Pocket PC Developer Network

PIE

Information about Pocket Internet Explorer and how to create sites/pages for Pocket Internet Explorer.

Articles at Pocket PC Developer Network

- Writing ASP applications for Pocket PCs
  November 06, 2001. Active Server Pages (ASP) has provided an excellent environment for developers to rapidly create stable, feature rich, data driven applications. PocketASP brings this power the Pocket PC, enabling developers to apply their existing skills to an exciting new platform. This article explains how to get from installation through to creating a working PocketASP application.
- Developing A New Outlook - PocketASP + POOM
  January 31, 2002. Active Server Pages (ASP) has provided an excellent environment for developers to rapidly create stable, feature rich, data driven applications. PocketASP brings this power to the Pocket PC, enabling developers to apply their existing skills to an exciting new platform.

MSDN Technical Articles

- What's New in Pocket Internet Explorer
  Get started using the new features in Microsoft Internet Explorer for the Pocket PC-I'll walk you through each feature using sample code.
- Launching Your Application from Pocket Internet Explorer
  September 12. Learn how to specify a "starter" file extension, so users can launch your application directly from the Web on their Pocket PCs.
- Pocket PC and Handheld PC Browser Comparison
  Two different browsers are designed for Microsoft WindowsR Powered Pocket PCs and Handheld PCs. Internet Explorer for the Pocket PC features a smaller software footprint and adaptability to Pocket PC's smaller screen. Internet Explorer version 4.02 for Handheld PC 2000 is a port of Internet Explorer 4.0 for the desktop PC. The functionality was an appropriate fit for the Handheld PC models' half and full screens.
- Designing Web Sites for the Internet Explorer for Pocket PC
  Jan. 31, 2001. One of the best new features of the Pocket PC is the new Internet Explorer Web brower for Pocket PC (hereafter called Pocket Internet Explorer) . For the first time in any hand-held device, Pocket Internet Explorer allows the Pocket PC owner not only to browse online Web content but also to synchronize Web pages for offline viewing. The intent of this white paper is to help Web designers and developers create Web sites that are compatible and optimized for viewing via Pocket Internet Explorer on the Pocket PC.
- Make Your Web Applications Support Pocket PC
  May 23, 2001. Make your current Web applications look better on Pocket PCs with simple measures like checking the client browser on the server side. The same logic can be used to also support MME clients.
- XSL ISAPI Filter 2.1: Server-side XSL Formatting for Multiple Device-types
  Feb. 1, 2001. Use XML and XSL to provide Web pages for PCs, cell phones (WML, HDML), and Pocket PCs find source code and samples.
- Microsoft XML Support with the Internet Explorer for Pocket PC
  Nov. 9, 2000. Download, view, and manipulate rich, structured data in XML format on any Pocket PC. Here's how.
- Is That a Script in Your Pocket?
  Aug. 14, 2000. Make your Web pages interactive in the Internet Explorer for Pocket PC through Microsoft JScript.

- Internet Explorer for Pocket PC - HTML and Object Model Reference
  May 9, 2000. Easy to use time-saving reference with HTML ready for you to include in your projects.
- Microsoft Internet Explorer and Web-based Applications for Pocket PC
  . This PowerPoint presentation introduces the technologies that make up the Pocket PC platform and covers their general capabilities.
- Programmer's Guide to Internet Explorer for Microsoft Windows CE 3.0
  June 2000. This white paper discusses the differences between the Windows CE-based and the Win32-based versions of Internet Explorer, and the technologies that are supported by Internet Explorer for Windows CE. (14 printed pages)
- Using XML and XSL in Pocket Internet Explorer
  August 14, 2001. You can use the support for XML, XSL, and scripting in Pocket Internet Explorer to create both online and offline business applications.
- Creating ActiveX Controls for the Internet Explorer on Pocket PC, via the Active Template Library
  Nov. 9, 2000. Learn how to build ActiveX Controls with ATL for Windows CE and use them in Microsoft Internet Explorer-based applications.

MS Knowledge base

- Q275230 - HOWTO: Specify a URL When Starting Pocket Internet Explorer from eVB
  11/30/2000. This article demonstrates how to start Microsoft Pocket Internet Explorer on a Pocket PC with a specific URL from eMbedded Visual Basic (eVB) code.
- Q296904 - HOWTO: Install the PocketPC Emulator with JScript Support
  6/2/2001. This article explains how to install and configure the Pocket PC Emulator that ships as part of Microsoft Embedded Visual Toolkit 3.0.

DEVBUZZ.COM

- PocketASP, ASP on your Pocket PC
  5 Dec 2001. I just love ASP - deVBuzz is an out an out ASP + SQL Server site. Hence no secret about why I loved this PocketASP article from Vince Singleton of ModeZero - the company behind PocketASP. "Give a lazy man the hardest job and he'll find the easiest way of doing it. Now I'm not saying I'm lazy of course, but when I first started looking into developing applications on the Pocket PC platform there's plenty there that makes you think there could be some long nights ahead."
- Creating POOM items using PIE Web pages
  29 Nov 2001. The ability to manipulating Pocket Outlook items such as Contacts, Appointments and Tasks from an eVB app is a very cool feature. Once the various items are added into the outlook database, your app could filter and display those items based on the user's preferences, or allow the user to quickly find a specific item. However, the process of creating a new outlook item is usually a manual process. The user typically selects "New" and then tediously enters the various text data of the item using the very small SIP keyboard or by writing on the screen. This is not only a time consuming task, but it is prone to errors.

Sections:
Audio
COM/ActiveX
Control Panel
Databases
Debug
Device Information
.NET CF
Email, POOM
Emulator
Enterprise
eVB
eVC++
Events
Files and Registry
GAPI
Hardware Buttons
HTML in Programs
Images
Installation
IrDA, LED
Misc
Network, Internet
Non-Programming
Password Protection
PIE
Pocket PC 2002
Power
Printing
SIP
Synchronization
Today
Tools
User Interface
XML

Writing ASP applications for Pocket PCs

By Paul Adams, November 05, 2001.
Print version

Introduction

Active Server Pages (ASP) has provided an excellent environment for developers to rapidly create stable, feature rich, data driven applications. PocketASP brings this power the Pocket PC, enabling developers to apply their existing skills to an exciting new platform. This article explains how to get from installation through to creating a working PocketASP application.

Installation

The evaluation version of PocketASP is available from http://www.ModeZero.net/PocketASP.

It is fully functional and not time limited. If machine space is available, it is recommended that the ADOCE 3.1 version be downloaded as that contains the most up-to-date ADO drivers.

On The PC

1. Download the zip file
2. Unzip to a temporary location on the PC
3. Open the readme.htm file for last minute information and the licence agreement
4. Copy the .CAB file onto your Pocket PC

On the Pocket PC

1. Close Pocket Internet Explorer if necessary
2. Open Explorer and click on the .CAB file just copied down from the PC
3. The required files will be copied and registered and the .CAB automatically removed

Now to test that the installation was successful:

1. Open Pocket Internet Explorer
2. If the Address Bar is not visible, click View->Address Bar
3. Click on the text in the address bar and type *pasp://ModeZero/*
4. The default homepage, "Welcome to PocketASP", will appear

Demonstration Pages

As a warm-up, before diving into creating your own pages, follow the link on the homepage to see the examples. Some of the key supported ASP objects and methods are demonstrated, such as:

- Processing forms
- Calling functions
- Using include files
- Database support
- Sessions
- Cookies

Development Environment

The PocketASP project enables ASP developers to create their web applications in the normal way (on a PC) and then adds a new final stage of copying the pages down to the Pocket PC when the development is complete. The caveat is to be aware of currently supported objects and methods, as described in the release notes. For example, persistent cookies are supported, but per-session cookies are not, which means using the Session object to store session-based information.

While developing on the PC, it's possible to get an approximation of how the pages will look on the Pocket PC device. Using IE5, with *View->Text Size* set to *Smallest*, the following JavaScript creates a suitably sized window:

```
window.open("http://Machine/PocketASP/MyDomain/Default.htm",
 "PocketPCView",
 "toolbar=no,location=yes,directories=no,menubar=no,scrollbars=yes,resizable=no,width=252,height=275");
```

Hello World Example

An introduction to a platform would not be complete without a Hello World application. The example below was developed using Visual Interdev and tested on a machine running IIS, before being copied down to an iPaq H3630 for final testing.

Here's a step-by-step guide to getting an ASP page up and running.

1. Write the ASP code

Below is an example of some over-engineered code to output a Hello World message.
Hello.asp

```
<html>
<body>
<%
        sHello = "Hello"
%>
<hr>
<p align="center">
        <%= sHello & " " & WorldFn %>
<p>
<hr>
<%
'----------------------------------------
function WorldFn()
        WorldFn = "World!"
end function
'----------------------------------------
%>
</body>
</html>
```

2. Test It on the PC

Place the Hello.asp file into a virtual directory under IIS. Lets call it FirstExamples.

Navigating to http://MyMachine/FirstExamples/Hello.asp should result in a friendly Hello World! message appearing in your browser.

3. Create a new Pocket PC pseudo-domain

The installation process creates a directory called *PAspPages* in the root of the Pocket PC. The directories contained in *\PAspPages* are treated as pseudo-domains (for example *ModeZero* is created by the install for the demonstration pages). Note that, just like real domains, the pseudo-domain names should not contain spaces.

We need to create a new directory (pseudo-domain) under *\PAspPages* for our Hello World example page. Again, we'll use *FirstExamples*.

4. Copy to Pocket PC

Use Explorer to find the recently created */PAspPages/FirstExamples/* directory on the Pocket PC and copy Hello.asp from the PC to that directory.

5. Test it on the PocketPC

To see the finished product, open Pocket IE and type *pasp://FirstExamples/Hello.asp*. The same friendly message will appear on your browser.

The Hello World example shows that there are no tricks or hidden steps, the source files don't get mangled or transformed in any way (in fact, you could develop and edit them on the device). The only alteration that may be needed is to the look and feel of your pages to make allowances for the amount of screen space available.

Database Access

Including database functionality in your Pocket PC application is, again, business as usual for the ASP developer. The database support is provided using ADOCE, which is a subset of ADO. The core functionality of ADO is provided, i.e. adding, updating and removing information from tables.

Below are the steps to take to get your ASP application ready for database access:

1. Design and create the database on the PC using Access
2. Copy the .MDB file into a suitable directory on the Pocket PC using ActiveSync
3. As part of the copy process ActiveSync will convert the database into a .CDB file
4. In addition to the conversion ActiveSync also offers to keep the master (PC) database and the Pocket PC databases in sync.
5. Write ASP pages using standard ADO connect and recordset calls.
(note that the DSN for connection is just the path to the .CDB file created.)

For example:

Connect to Pocket PC Database

```
set oDBCon = Server.CreateObject("ADODB.Connection")
oDBCon.ConnectionString = "data source = \PAspPages\ModeZero\_db\TestCEDB.cdb"
oDBCon.Open
```

Open a Recordset

```
Set rs = Server.CreateObject("ADODB.RecordSet")
rs.Open "PersonalInfo", oDBCon, 1, 3
```

(taken from the DemoDB.asp sample page)

Sessions And Cookies

If your application needs to store per-session or cross-session information then the Session and Cookie objects and methods are available and function in the usual way. The cookies use the pseudo-domain described above to define their scope, for example cookies in the *ModeZero* domain cannot be accessed from the FirstExamples domain.

Releasing Your Application

Finally, when it comes to releasing your application, developing in ASP means you are free from the current headache of compiling and maintaining versions over the different flavours of Pocket PC processor. The same code will run on all supported PocketASP platforms, making development and testing times shorter and source control simpler.

Summary

By using their existing skills, ASP developers can quickly create exciting new applications across the Pocket PC platforms. The database synchronisation features already provided by ActiveSync 3.1, coupled with the database support in PocketASP, enable access to a wealth of new data driven Pocket PC application opportunities. For many Pocket PC applications, developing in ASP will be quicker, more stable and easier to rollout than using the traditional alternatives.

Resources

Information on PocketASP can be found at http://www.ModeZero.net/PocketASP. There is also a discussion group on Yahoo! at http://groups.yahoo.com/group/PocketASP.

Related resources:

- Section: PIE
- Article: Developing A New Outlook - PocketASP + POOM
- Article: Windows CE Web Server

Discuss

Discuss this article. Here you can write your comments and read comments of other developers.

Rate this article:
Poor ○ ○ ○ ○ ○ Excellent Rate
1 2 3 4 5

Sections:
Audio
COM/ActiveX
Control Panel
Databases
Debug
Device Information
.NET CF
Email, POOM
Emulator
Enterprise
eVB
eVC++
Events
Files and Registry
GAPI
Hardware Buttons
HTML in Programs
Images
Installation
IrDA, LED
Misc
Network, Internet
Non-Programming
Password Protection
PIE
Pocket PC 2002
Power
Printing
SIP
Synchronization
Today
Tools
User Interface
XML

Pocket PC Developer Network / Articles /

Developing A New Outlook - PocketASP + POOM

By Ben Gladwyn, January 31, 2002.
Print version

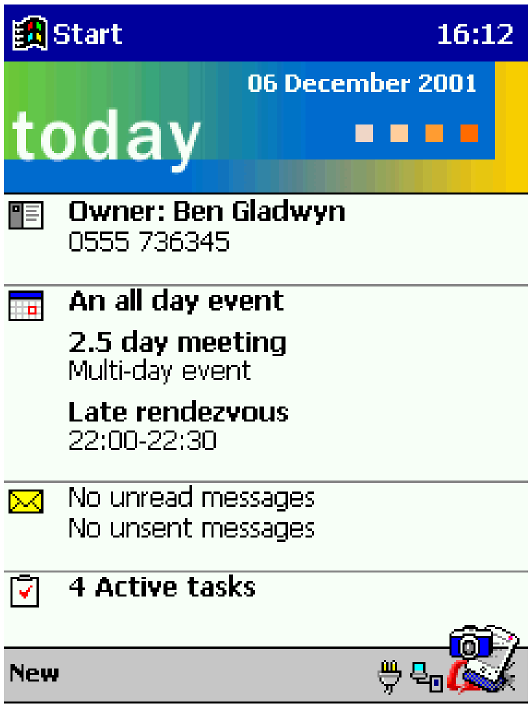
Introduction

Active Server Pages (ASP) has provided an excellent environment for developers to rapidly create stable, feature rich, data driven applications. PocketASP brings this power to the Pocket PC, enabling developers to apply their existing skills to an exciting new platform.

In this article an example application is created that provides a replacement for the "Today" view using the Pocket Outlook Object Model. All source code for may be downloaded here (12 Kb).

Replacing the Today view

Here is a typical Today view on my iPaq. (All screen shots have been captured using IA ScreenShot).



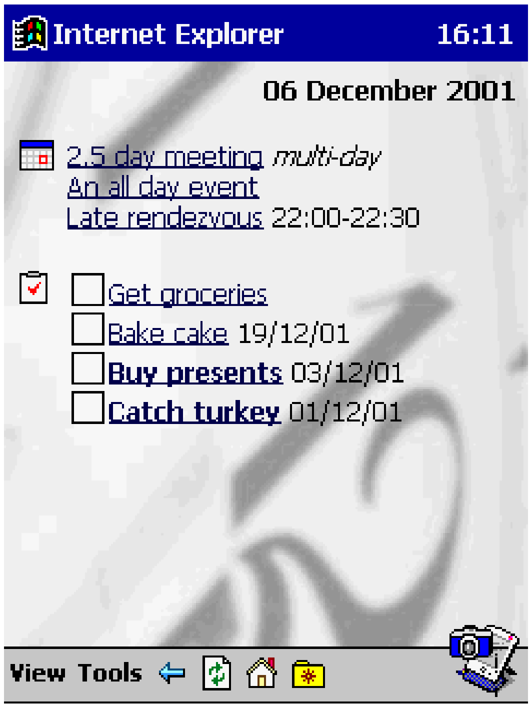
Owner: Ben Gladwyn
0555 736345

An all day event
2.5 day meeting
Multi-day event
Late rendezvous
22:00-22:30

No unread messages
No unsent messages

4 Active tasks

I'd like a replacement view that lists today's active tasks, as well as making better use of precious screen real estate. Perhaps something like this:


Internet Explorer 16:11

06 December 2001

2.5 day meeting *multi-day*
An all day event
Late rendezvous 22:00-22:30

Get groceries
Bake cake 19/12/01
Buy presents 03/12/01
Catch turkey 01/12/01

Opening the Pocket Outlook Application object

The Pocket Outlook Application object is the primary interface to POOM. One of these objects has to be created and logged in to before any other calls are made.

At the end of each page, there should be call to polApp.Logoff().

I decided to prevent duplication of creation and logon code in each ASP page by accessing the application object through the following function,

```
' ---------------------------------------------------------------------------
' ---------------------------------------------------------------------------
' polApp: Hides complexity of creating object and logging on from rest of
'         application
'
Dim f_polApp
Function polApp
        ' Check to see if object has already been created.
        ' If it has, return it.
        ' If it hasn't, create it, log in, and return it.
        '
        If IsEmpty(f_polApp) Then

                Set f_polApp = CreateObject("PocketOutlook.Application")
                f_polApp.Logon()
        End If

        Set polApp = f_polApp
End Function
' ---------------------------------------------------------------------------
```

Listing the incomplete tasks

To get a list of Outlook items via POOM we use the getDefaultFolder method, passing in the ID of the information type we querying (the folder identities and other constants are defined in an included file). A collection of the items in that folder can then be filtered and sorted before display as required.

To get the tasks folder and limit it to incomplete tasks only we use,

```
' Get the tasks folder and then access the Items collection within it
'
Set taskfolder = polApp.getDefaultFolder(olFolderTasks)
Set taskitems  = taskfolder.Items

' Limit to incomplete tasks only
'
taskquery = "[Complete] = False"
Set taskitems  = taskitems.Restrict(taskquery)

' Check to see if any incomplete tasks returned
'
if (taskitems.Count > 0) then

  ' Sort collection by due date (ascending)
  '
  taskitems.Sort "[DueDate]", False

  ' Iterate through the collection
  '
  for each task in taskitems

    ' Output subject
    '
    %>
    <%=htmlEncode(task.Subject)%><br>
    <%

  next

end if
```

(The htmlEncode() function takes a string and converts it to HTML so it displays correctly in the browser.)

Task completion and due date

Using the Item interface, we can add, view, modify and delete entries in Outlook. The first thing to do is to create a mechanism to mark tasks as being complete.

```
' Set a task to be complete
'
if (request.Form("f_action") = "completetask") then

  ' Get individual task
  '
  Set task = polApp.GetItemFromOid(request.Form("oid"))

  ' Set it to be completed
  '
  task.Complete = true

  ' Save it
  '
  task.Save

end if
```

The task list generated is now rendered as a form with checkboxes. As soon as a tick is placed, the page reloads and sets the task in question to be complete.

While we're changing the task list, we can query the DueDate property. If one exists then we'll display it and if it's in the past we change the emphasis of the subject.

```
' Check to see if any incomplete tasks returned
'
if (not IsNull(taskitems)) then

  ' Create top of form
  '
  %>
  <form method="post">
  <input type="hidden" name="f_action" value="completetask">
  <%
```

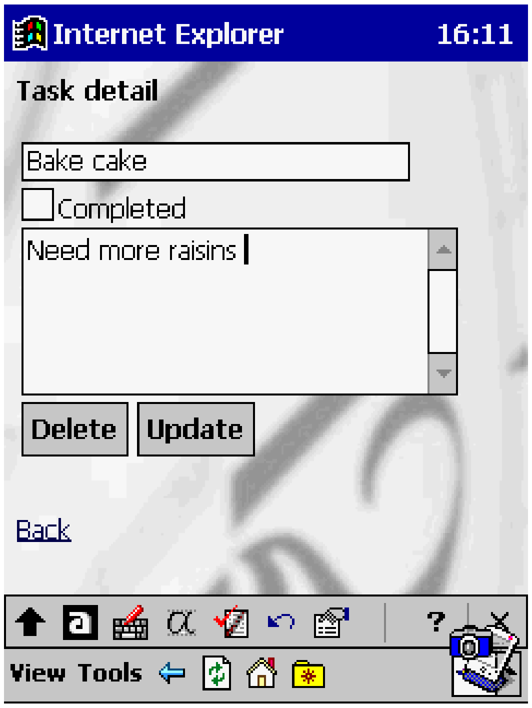
```
   ' Iterate through the collection
   '
   for each task in taskitems

     ' Use this boolean when formatting data
     '
     taskoverdue = false

     ' Check for empty duedate
     '
     if (task.DueDate = #1/1/4501#) then

       taskdate = ""

     else

       ' Format duedate for output
       '
       taskdate = formatDateTime(task.DueDate, 2)

       ' Check to see whether task is overdue
       '
       if (task.DueDate < Now) then taskoverdue =
true
     end if

     ' Output checkbox, subject and duedate
     '
     %>
     <input type="checkbox" name="oid" value="<%=task.Oid%>" onClick="submit()">

     <%if (taskoverdue) then Response.Write "<b>"%>
     <%=htmlEncode(task.Subject)%>
     <%if (taskoverdue) then Response.Write "</b>"%>

     <%=taskdate%>
     <br>
     <%

   next

   ' Close form
   '
   %>
   </form>
   <%

end if
```

Note the use of the literal date #1/1/4501#. This value is used by Outlook when a date field is empty.

Task view, update and delete

Each item in the Outlook information store is uniquely identified by an assigned object ID (OID). This applies to appointments, tasks and contacts. Once we have the OID of a task, we can query its contents, alter those contents and delete it.

If the task data is displayed in a pre-populated HTML form, updates can be applied and committed quickly.



```
' Get OID from request
'
oid = request.QueryString("oid")
if (oid = "") then oid = request.Form("oid")

' Must have an OID
'
if (oid <> "") then

  ' Get individual task
  '
  Set task = polApp.GetItemFromOid(oid)

  ' Is this task to be deleted?
  '
  if (request.Form("f_action") = "delete") then

    ' Delete the task
    '
    task.Delete

    ' Return to home page
    '
    response.Redirect("default.asp")

  end if

  ' Is the information to be updated?
  '
  if (request.Form("f_action") = "update") then

    ' Update task details
    '
    task.Subject  = request.Form("taskSubject")
    task.Body     = request.Form("taskBody")

    if (request.Form("taskComplete") = "1") then
      task.Complete = True
    else
      task.Complete = False
    end if

    ' Save these changes
    '
    task.Save

  end if
  ' Display HTML form
  '
  %>
  <p>
  <form method="post" name="taskForm">
  <input type="hidden" name="f_action" value="update">
  <input type="hidden" name="oid" value="<%=task.Oid%>">
  <input type="text" name="taskSubject" value="<%=htmlEncode(task.Subject)%>"><br>
  <input type="checkbox" name="taskComplete" value="1"
  <%
  if (task.Complete) then response.write(" checked")
  %>
  >Completed<br>
  <textarea name="taskBody"><%=htmlEncode(task.Body)%></textarea><br>

  <input
   type="submit"
   onClick= "
   if (confirm('Are you sure you want to delete this task?'))
   {
     document.taskForm.f_action.value='delete';
     return true;
   }
   else
   {
     return false;
   }
   "
   value="Delete"
  > | <input type="Submit" value="Update">
  </form>

  <p><br><br><a href="default.asp">Back</a>
  <%

else

  ' Empty OID => return to home page
  '
  response.Redirect("default.asp")

end if
```

That's all our task functionality complete. More features could be added, like creating new tasks, making more fields editable or displaying different icons depending on the task status/type. However we'll move on and modify the code we've already created to list and edit appointments.

Listing today's appointments

With a few modifications to the task list code, we can now list today's appointments. The restriction placed on the Items collection is more complex. We need to list all appointments that are either all day events for this day, timed events for today which are not in the past and current timed events which span multiple days.

```
' Get the calendar folder and then access the Items collection within it
'
Set calfolder = polApp.getDefaultFolder(olFolderCalendar)
Set calitems  = calfolder.Items

nowdate  = FormatDateTime(Now,2)
nowtime  = FormatDateTime(Now,4)
calquery = ""

' Limit to today's appointments only
'
' 1. all day events
calquery = calquery + "("
calquery = calquery + "[AllDayEvent] = True"
calquery = calquery + " and [Start] <= """ + nowdate + """"
calquery = calquery + " and [End] >= """ + nowdate + """"
calquery = calquery + ")"
```

```
' 2. Timed events
calquery = calquery + "or ("
calquery = calquery + "[AllDayEvent] = False"
calquery = calquery + " and [Start] <= """ + nowdate + """"
calquery = calquery + " and [End] >= """ + nowdate + " " + nowtime+ """"
calquery = calquery + ")"

Set calitems  = calitems.Restrict(calquery)
```

When listing the appointments we need to display different information, depending on its type. If it's a normal appointment (for today, at a specified time) we should list the start and end times. If it's an all day event, we don't need to put anything. If it's a timed appointment which spans across multiple days, we'll put "multi-day" next to its listing - just like in the Today view.

```
' Sort collection by Start time (ascending)
'
calitems.Sort "[Start]", False

' Iterate through the appointment list
'
for each appointment in calitems

  ' Use this boolean when formatting data
  '
  allday = appointment.AllDayEvent

  appointmentstart = ""
  appointmentend   = ""
  appointmenttimes = ""

  if (not allday) then

    ' Does this appointment start today ?
    if (appointment.Start >= DateValue(nowdate)) then
      appointmentstart = FormatDateTime(appointment.Start,4)
    end if

    ' Does this appointment end today ?
    if (appointment.End < DateAdd("d",1,DateValue(nowdate))) then
      appointmentend = FormatDateTime(appointment.End,4)
    end if

    if (appointmentstart = "" and appointmentend = "") then
      appointmenttimes = "<i>multi-day</i>"
    else
      appointmenttimes = appointmentstart + "-" + appointmentend
    end if
  end if

  ' Output checkbox, subject and duedate
  '
  %>
  <a href="app.asp?oid=<%=appointment.Oid%>"><%=htmlEncode(appointment.Subject)%></a>
  <%=appointmenttimes%><br>
  <%
next
```

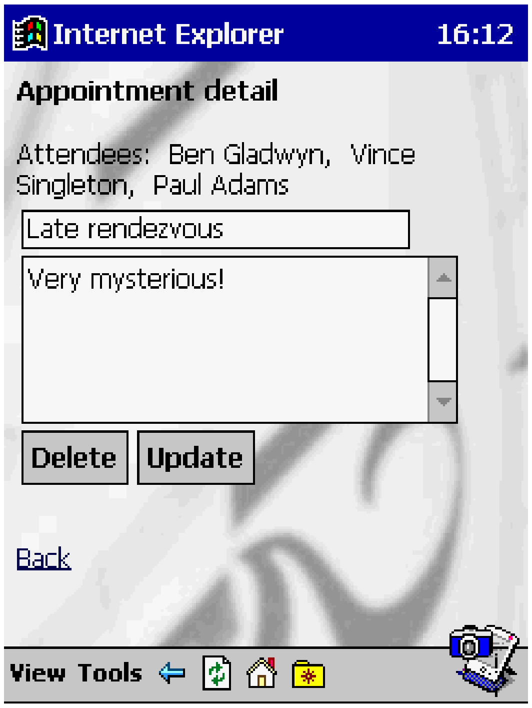
Appointment view, update and delete

We can use the task detail code above as the basis for our appointment detail code. By removing code that operates on task.Complete and global replacing "task" with "appointment" we have a functional page that displays and updates the subject and body of an appointment.

It would be useful to see who the attendees for a particular meeting are. This can be done easily by looping through the Recipients collection of the appointment.

```
' Get attendee collection for this appointment
'
set appAttendees = appointment.Recipients

' Check to see if there are any attendees
```

```
'
if (appAttendees.Count > 0) then

  %>Attendees: <%

  ' Loop through collection and render the name of each of the attendees
  '
  for count = 1 to appAttendees.Count

    set recip = appAttendees.Item(count)
    %>&nbsp;<%=htmlEncode(recip.Name)%><%

    if (count < appAttendees.Count) then Response.Write(", ")
  next

  %><br><%

end if
```



Conclusion

Attractive Pocket PC applications that need to access Outlook data can be built quickly using the combined functionality of POOM and PocketASP.

The simple application created here could be expanded to query more of the task and appointment properties, as well as linking through to the contact details of an attendee. Going further, additional information could be extracted from a Pocket Access database to augment the data already held in Outlook. Web programming skills go a lot further than they used to with PocketASP!

Related resources:

- Section: Email, POOM
- Section: PIE
- Article: Writing ASP applications for Pocket PCs
- Library: Pocket Outlook .NET Wrapper
- Article: Windows CE Web Server
- Article: Using eMbedded Visual Basic to Retrieve Pocket Outlook Contacts
- Article: Using the Pocket Outlook Object Model SDK
- Article: Implementation of POOM Using Embedded Visual C++ 3.0

- Article: Incorporating Pocket Outlook Data into Your Microsoft .NET Compact Framework-based Applications

Discuss

Discuss this article. Here you can write your comments and read comments of other developers.

Rate this article:
Poor ○ ○ ○ ○ ○ Excellent Rate
1 2 3 4 5

What's New in Pocket Internet Explorer

Microsoft Corporation

June 2002

Applies to:
Microsoft® Windows® Powered Pocket PC 2002

Summary: Get started using the new features in Microsoft Internet Explorer for the Pocket PC and step through each feature using sample code. (6 printed pages)

What you need

- Your current Web application tool set (Microsoft® Visual InterDev®, Microsoft FrontPage®, and so forth).
- A live Internet connection from your Pocket PC is helpful.

Contents

An Enhanced Browser

With the introduction of Pocket PC 2002, there have been even more exciting changes and new features available in Pocket Internet Explorer. A vast array of features awaits any Web developer who looks deep under the covers of what may at first seem like the "same old browser."

When designing Web applications for Pocket PC, you will mainly have to think about:

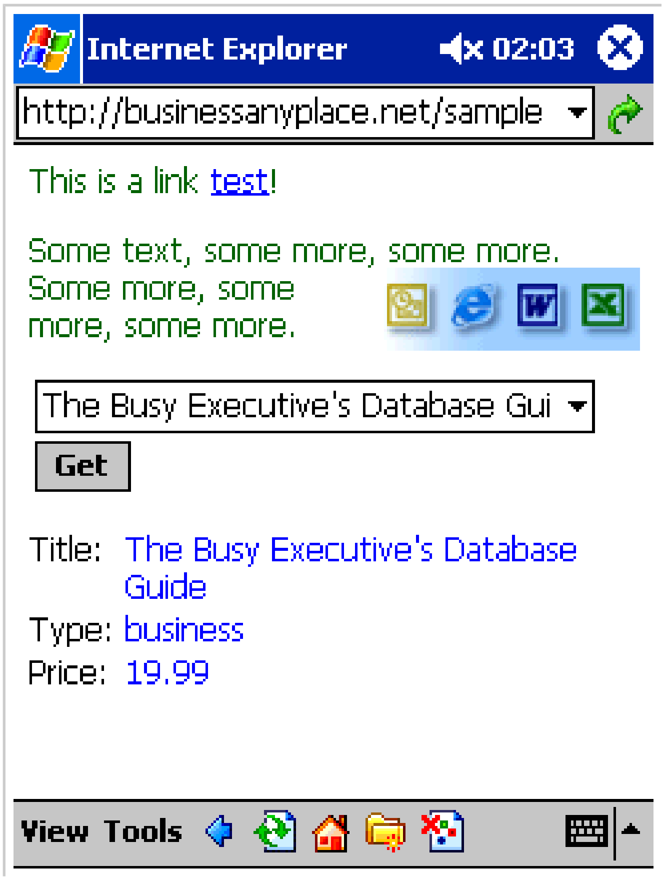
- Bandwidth
- Screen size
- Features

A general rule is to create pages that have a maximum width of 220 pixels since this will save the user from horizontal scrolling. In a Web application meant to be used over a wireless wide area network, bandwidth is really an issue. This means that you should aim for small file size (for shorter load time) when you design your Web pages. As user design research has shown, usability increases as the interface gets—less is more!

Designing for a smaller screen will help you keep the size of your pages down (it will even make you create smaller images).

A Sample Page

Now, let's look at the sample HTML pages. If you use your Pocket PC to read this article, you can find the sample online.



This is a link test!

Some text, some more, some more. Some more, some more, some more.

The Busy Executive's Database Gui

Get

Title: The Busy Executive's Database Guide

Type: business

Price: 19.99

Figure 1. Sample "interactive" HTML page

And here is the source code for this page:

```
<html><head><title>Test</title></head>
<body vlink="purple" alink="red">
<basefont color="darkgreen">
This is a link <a HREF="test.asp">test</a>!
<br><br>
Some text, some more, some more.<img
src="toolbar.gif" width="100" height="31" border="0" align="right">Some
more, some more, some more.
<br><br>
<xml id="xmlTitles" src="http://www.businessanyplace.net/pubs?SQL=select+*+from
+titles+for+xml+auto&root=titlelist"></xml>
<span id="divTitleCombo"></span>
<input type="button" value="Get" onclick="javascript:loadTitle();">
  </tr>
</table>
<br><br>
<table cellspacing="0" cellpadding="0" border="0">
  <tr>
    <td valign="top">Title: </td>
    <td valign="top"><span id="divTitleName"></span></td>
  </tr>
  <tr>
    <td valign="top">Type: </td>
    <td valign="top"><span id="divTitleType"></span></td>
  </tr>
  <tr>
    <td valign="top">Price: </td>
    <td valign="top"><span id="divTitlePrice"></span></td>
  </tr>
</table>
<br>
<script for="window" event="onload">
  var selectHTML = "<select name=\"cboTitle\" size=\"1\">\n" +
        "<option value=\"0\" selected>Select title</option>\n";
  var root = xmlTitles.documentElement;
  var listTitles =  root.selectNodes("//titlelist/*");
  var nodeTitle;
  var i;
  for (i = 0 ; i < listTitles.length; i++) {
    nodeTitle = listTitles.item(i);
    selectHTML += "<option value=\"" +
      nodeTitle.attributes.getNamedItem("title_id").value + "\">" +
      nodeTitle.attributes.getNamedItem("title").value + "</option>";
   }
  selectHTML += "</select>\n";
  divTitleCombo.innerHTML = selectHTML;
</script>
<script>
void function loadTitle() {
  if (cboTitle.value == "0") {
    alert("Select a title in the list!");
    return;
  }
  var root = xmlTitles.documentElement;
  var nodeTitle =  root.selectSingleNode("//titles[@title_id=\"" +
                   cboTitle.value + "\"]");
  divTitleName.innerHTML = "<font color=\"blue\">" +
    nodeTitle.attributes.getNamedItem("title").value + "</font>";
  divTitleType.innerHTML = "<font color=\"blue\">" +
    nodeTitle.attributes.getNamedItem("type").value + "</font>";
  if (nodeTitle.attributes.getNamedItem("price") != null)
    divTitlePrice.innerHTML = "<font color=\"blue\">" +
      nodeTitle.attributes.getNamedItem("price").value + "</font>";
  else
    divTitlePrice.innerHTML = "<font color=\"blue\">?</font>";
}
</script>
</body></html>
```

Improved HTML

First, you may notice that we have included a vlink and alink attribute to the <body> tag. The HTML above will make visited links appear in purple and active links appear in red.

A welcome addition to the image <img> tag is the align attribute. You can align images to the left, middle, or right. In the code above you can see that I have aligned the image to the right, making the text "float" on the left. If I want to break the text around the image, the line-break tag now has the standard clear attribute. The align attribute has been added to all these tags: <img>, <table>, <caption>, and <div>. In addition, you can now set the color of the normal text with the color attribute on the <basefont> tag. In this sample, I have set the color to dark green.

Extended DOM

You can see in the sample code above that the DOM (Document Object Model) is extended. The most important addition is the ability to use the innerHTML and innerText properties on the

and tags. This makes it possible for you to create HTML dynamically (hence the name Dynamic HTML, or DHTML).

In the example above, I have used this technique to show individual titles in the standard pubs database. I have also loaded an XML (Extensible Markup Language) file and used the XML DOM to manipulate it. As you can see, the XML loads directly from a Microsoft SQL Server™ 2000 using its XML support. If you don't have a connected Pocket PC, put the sample files on your device and just replace the src attribute on the <xml> tag with titles.xml. Other interesting additions to the DOM include the screen object and the ability to use window.open to allow "user interaction" to open those script-generated "navigates."

The final Pocket PC 2002 SDK documentation will include a full reference for the HTML and Microsoft Jscript® capabilities of Pocket Internet Explorer, as well as information on detecting the browser and a style guide for authoring content for the browser.

Wireless Application Protocol

If you're still stuck in the dark ages of WAP (Wireless Application Protocol), you should be happy to see that it's now supported. This is actually something that you don't find in the PC version of Internet Explorer. The WAP support includes WML (Wireless Markup Language), WMLScript, and WBMP (Wireless BitMaPs). There is also support for WSP (Wireless Session Protocol) that enables compression of HTTP headers to minimize demand on bandwidth.

Web Server

There will be a Pocket PC Web server in the final SDK. With it, you can run your Web-based applications offline—even using Microsoft ASP (Active Server Pages). You can also use http://localhost in Pocket Internet Explorer to access the local Web server.

Always Want More

I welcome all these new features, but as always I have a couple of bullets on my wish list. Things like support for CSS (Cascading Style Sheets), client-side VBScript, and more extensive DHTML support would improve Pocket Internet Explorer even more.

For the Mobile Web Developer

If you want to deploy a custom application on your users' Pocket PCs, you are probably interested in some of the enhancements when developing client-side, Web-aware applications. I will briefly cover just some of the more important improvements, but each topic is probably material for a separate article.

HTML Control

The HTML control now includes a DTM_NAVIGATE message that essentially allows a developer to wrap the functionality of Pocket Internet Explorer inside a custom application. The HTML Control now supports downloading of images, sounds, and other "external" content, automatically. For example, simply return "0" for NM_INLINE_IMAGE and the HTML control will download the image, as shown in the sample HTMLHost included in the SDK.

New URL Moniker APIs

With the new URL moniker APIs you can get better control when interacting with the Internet using HTTP or WAP. Some new APIs at your disposal are URLOpenPullStream, which is used to pull down chunks of content from the Internet, and URLDownloadToFile, which is used to download from an URL to a local file. Others are URLDownloadToCacheFile, URLOpenBlockingStream, and URLOpenStream.

XML Parser

The XML Parser helps to manipulate XML without building an in-memory tree representation of the entire XML document. This interface was designed for developers who want to scan a chunk of XML to find a certain tag. The Push Model XML Parser includes an IXMLParser interface that parses XML either from an IStream, or directly from an in-memory buffer or a URL.

Events in ActiveX Controls

Microsoft ActiveX® controls in Pocket Internet Explorer now support events. This requires that ActiveX controls implement the IProvideClassInfo interface. ATL controls provide a default implementation of this interface, but it must be manually added to MFC-based controls.

ActiveScript Hosting

With the IActiveScriptHost interface you can use the JScript engine from within your own application.

Context Menu Extensions

This will enable you to add context-sensitive menu options to Pocket Internet Explorer (when you tap-and-hold on a Web page).

Conclusion

Pocket Internet Explorer provides a capable platform for Web and browser-based applications when the Pocket PC is connected to the Internet. Security options, such as 128-bit SSL or accessing intranet sites via the virtual private networking features, allow you to deploy professional business Web applications for the Pocket PC. You just have to get going!

For Business

Assess the Business Value of Windows Mobile Software

Companies and organizations worldwide are implementing mobile business solutions to accelerate business cycles, increase productivity, reduce operating costs and extend their enterprise infrastructure.

- Why Windows Mobile software?
- Four factors making mobile devices smart business
- Collaboration solutions for a mobile workforce
- New book now available: Enterprise Guide to Gaining Business Value From Mobile Technologies

Windows Mobile and Insurance

Microsoft and HP Enable Insurance Claims Adjusters with Mobile Solution

Learn how the HP Mobile Agent can help the insurance industry. HP Mobile Agent enables agents to close business more quickly and effectively at the point-of-sale, and adjusters to complete claims reports in real time, when they are onsite with the insured.

- Download the .pdf

Resources to Build and Deploy

Windows Mobile Solution Providers

Whether you're looking to mobilize your sales force, secure your mobile data, or create greater efficiencies within your organization, Solution Providers can help.

- Learn more
- Search Solution Providers database
- Search Security Product Solutions database

SMS 2003 Device Management Feature Pack

The Microsoft Systems Management Server (SMS) 2003 Device Management Feature Pack (Beta 2) helps SMS 2003 manage devices running Windows CE 4.2 or Windows Mobile 2003 software for Pocket PCs.

- Learn more

Windows Mobile Solutions Partner Program

We help mobile application developers worldwide build and bring to market innovative wireless solutions for Windows Mobile devices.

- Learn more

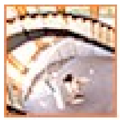
Upcoming Events

3GSM World Congress 2005

February 14-17, 2005, Cannes, France
Microsoft has an exciting program of events for 3GSM World Congress 2005, including many activities that ensure maximum interaction between our customers and partners with Microsoft representatives and senior executives.

- Learn more about 3GSM World Congress 2005

Microsoft Products and Technologies At Work



At Work: Discover Ways to Work Smarter

Find resources, tools, and other information to help you get more done and work more effectively whether you're in the office, at home, or on the road.

- Learn more

Essentials

White Papers

Evaluate how investments in mobile solutions can help your business.

- Supporting Windows Mobile-Based Devices Within the Enterprise
- Windows Mobile-Based Smartphones: Improving Business Productivity
- Making the Case for Wireless Mobility Investment (PDF)
- See all White Papers

Case Studies

Businesses share their deployment experiences.

Demonstrations

See mobile business solutions in action.

Books

Enterprise Guide to Gaining Business Value From Mobile Technologies

The Data Recovery Difference

"If you lose your Windows Mobile-based Smartphone, the data for the most part can be quickly recovered from your PC. Windows Mobile-based Smartphones allow us to manage all the information we used to keep on Day-timers in a single, handheld device. And it's not the end of the world if you lose it."
Bill, Salesperson, October 23, 2003

Designing Web Sites for the Internet Explorer for Pocket PC

White Paper

Applies to:
Microsoft® Windows® Powered Pocket PC 2000
Microsoft Pocket Internet Explorer

Abstract

One of the best new features of the Pocket PC is the new Internet Explorer Web brower for Pocket PC (hereafter called Pocket Internet Explorer) . For the first time in any hand-held device, Pocket Internet Explorer allows the Pocket PC owner not only to browse online Web content but also to synchronize Web pages for offline viewing.

The intent of this white paper is to help Web designers and developers create Web sites that are compatible and optimized for viewing via Pocket Internet Explorer on the Pocket PC.

Introduction to Pocket Internet Explorer

Welcome to the World of Pocket Internet Explorer!

With the release of the Pocket PC, Microsoft® Pocket Internet Explorer offers the richest Web experience of all comparable pocket-sized devices.

Pocket PCs can display rich content. Various Pocket PC models are offered with 4,096 or 65,535 color displays as well as 4 and 16 levels of gray scale.

Pocket Internet Explorer has implemented key Internet technology standards: HTML 3.2; Secure Sockets Layer (SSL) versions 2.0 and 3.0 for secure transactions; Microsoft JScript® for scripting Web page behavior; cookies for visitor tracking and a user's easier return to sites that require user authentication; and frames for formatting.

Pocket Internet Explorer even supports ActiveX controls that already reside on the Pocket PC.

Pocket Internet Explorer supports the key technology XML (Extensible Markup Language), which will enable people to more easily deploy business Internet applications to the Pocket PC.

Some of the innovative ways that Pocket Internet Explorer for the Pocket PC delivers the Web to the small screen include:

- The Shrink-to-fit capability, which dynamically resizes a Web site to maximize viewing on the smaller, vertically oriented screens of Pocket PCs.
- The auto-state detection determines automatically if the device is connected to the Internet and, if not, seamlessly diverts the browser to a cached version of the Web page.

Due to its unique design and the optimized mixture of Internet Explorer 3.02 (HTML Engine), Internet Explorer 4.0 (Scripting Engine) and Internet Explorer 5.0 (XML engine) we wrote this white paper to help Web site designers optimize their sites for Pocket Internet Explorer.

Sniffing for Pocket Internet Explorer on Your Server

Before we get into designing Web pages for Pocket Internet Explorer we have to cover one very important aspect…

How can a Web Server Determine if a Pocket PC Links to the Site?

If you are using Microsoft Internet Information Services 4.0 or later you will find a file named BROWSCAP.INI in the directory \WINNT \system32\inetsrv. This file contains descriptions of all known browsers at the time you installed the latest service pack.

Here is the description of the Pocket Internet Explorer you have to add to the BROWSCAP.INI:

```
; Pocket PC (aka Rapier)
[Mozilla/2.0 (compatible; MSIE 3.02; Windows CE; 240x320)]
browser=Pocket IE
version=4.0
majorver=#4
minorver=#0
platform=Windows CE
width=240
height=320
cookies=TRUE
frames=TRUE
backgroundsounds=TRUE
javaapplets=FALSE
javascript=TRUE
vbscript=FALSE
tables=TRUE
activexcontrols=TRUE
```

Pocket Internet Explorer is actually a mixture of Internet Explorer 3.02 (HTML), Internet Explorer 4.0 (Scripting) and Internet Explorer 5.0 (XML) components. That is why it will be identified as Microsoft Internet Explorer 3.02 while it uses version 4.0 inside the properties.

When Pocket Internet Explorer sends a request to your HTTP server, the following specific information is included in the HTTP request header:

```
UA-pixels: {i.e. 240x320}
UA-color: {mono2 | mono4 | color8 | color16 | color24 | color32}
UA-OS: {Windows CE (POCKET PC) - Version 3.0}
UA-CPU = {i.e. MIPS R4111}
```

Using the following server side script (ASP) lines you can now create special optimized pages as soon as a Pocket Internet Explorer enters your site:

```
'Check for Windows CE (Pocket PC, Palm-size PC, Handheld PC, Handheld PC Pro)
if (InStr(Request.ServerVariables("HTTP_USER_AGENT"), "Windows CE")) then
        { add Windows CE specific code }
else
        { add code for other platforms }
end if

'Check for Pocket PC
if (InStr(Request.ServerVariables("HTTP_UA_OS"), "POCKET PC")) then
        { add Pocket PC specific code }
else
        { add code for other platforms }
end if
```

To identify the Pocket Internet Explorer using client-side scripting (JScript) you can use the following code:

```
        var strNav = navigator.userAgent;
// Check for Windows CE (Pocket PC, Palm-size PC, Handheld PC, Handheld PC Pro)
        var isCE = strNav.indexOf("Windows CE");
        if(isCE > -1) {
                //add Windows CE specific code
        }
        else {
                //add code for other platforms
        }

// Check for Pocket PC
        var isPPC = strNav.indexOf("240x320");
                if(isPPC > -1) {
                 // add Pocket PC specific code
        }
        else {
                // add code for other platforms
        }
```

Supported HTML Tags

Pocket Internet Explorer is HTML 3.2 compliant, with some minor exceptions. Therefore all HTML Tags that are defined by this standard can be displayed in Pocket Internet Explorer. There are some limitations and issues of Pocket PC that a Web designer should be aware of:

- DHTML not supported
- Frames always have a border and can always be resized
- CSS are not supported

Here are some general guidelines for using HTML tags targeting Pocket Internet Explorer:

General Tag Issues

Issues with the TARGET Attribute of <A> tag
Pocket Internet Explorer cannot spawn multiple windows. Therefore if you use "target=_new" with the <A> tag it does not open a second window as your browser on the desktop would do. You can use the "target" attribute to point to a named frame Th <AREA>, <BASE> and <FORM> tags also support this attribute. Using any other specified target (i.e. "Target=_new", "Target=foo") will cause Pocket Internet Explorer to behave the same as if you have not specified a target at all.

While Pocket Internet Explorer does support _top and _parent, it does not support _self, _blank. If the target is something other than a named frame that already exists or a supported special value, it's the same as if the TARGET attribute were omitted completely. In this case the browser will navigate the frame/window where the link was tapped.

The browser supports the TARGET attribute on AREA, BASE, and FORM tags as well.

Fonts
Pocket PC includes four fonts:

- Tahoma (default font for variable width fonts)
- Bookdings
- Frutiger Linotype
- Courier (default fixed width font)

All other font faces are converted to the closer of those four fonts defined by their font description. Using the <PRE> tag for any fixed width content will ensure that Pocket Internet Explorer chooses the correct fixed width font.

Frames
Frames consume a lot of space on the screen just for the borders and the margins and therefore are generally not recommended for Pocket Internet Explorer. If you really must use frames, limit them to no more than two per screen.

Match up tags
Design your pages in a way that the HTML tags match up correctly.
In example:

```
Wrong: <TABLE><FORM>….</TABLE></FORM>
Correct: <TABLE><FORM>….</FORM></TABLE>
```

Incorrect matching can lead to unpredictable results in Pocket Internet Explorer.

BGSOUND Attribute
Background sounds are supported by Pocket Internet Explorer and can provide cool special effects. However only WAV files are supported and they usually take up a lot of space. Use sounds sparingly and limit your effects to a minimum. Since the LOOP attribute is not supported, you cannot create or use permanent background sounds.

Form Fields and Buttons

Text Fields and Text Areas
Pocket Internet Explorer renders text fields and text areas no wider than the width of the Pocket PC screen. The user might have to scroll horizontally to get to the field but it will always fit on the screen. If you have to use text fields and you want to achieve a clean look and good feel, design them from the beginning in a way that they do not exceed the width of the Pocket PC screen. Position long fields and other areas at the beginning of line.

Buttons
A button is a rectangular graphical object and is named with text that describes the action performed when tapped. Oftentimes buttons are named "submit" or "cancel." Buttons will follow the flow of the text and will scroll with the page. Ideally, buttons should be designed and placed in such a way that they are consistent, both within a particular page and across other pages. There is a limited amount of screen space for the placement of buttons, so use your judgment when deciding how many buttons to include on the screen. Buttons cannot submit any forms offline except when AvantGo synchronization is used.

They can be used to fire off events that a custom event handler can catch. These event handlers will work offline. As mentioned before with text fields, buttons and text areas are not rendered wider than the screen width of the Pocket PC.

Tables

The use of tables to display information can greatly enhance the way users view certain data. Use the WIDTH attribute to set the size of the table in the window. The best way to control the table size is to use the pixel values for the WIDTH attribute. Nested tables are supported.

If you choose to leave out the WIDTH attribute, the table renders according the following rules:

- If the Fit to Screen option is selected, the table will perfectly for to the screen width.
- If Fit to Screen is not selected, Pocket Internet Explorer uses a virtual 640x480 screen and renders the table to a width of 640 pixels.

Thus, unless there is a specific reason for having table or cell widths, it is recommended that width values be omitted. The ALIGN attribute is supported on the <TR> and <TD> tags, but not on the <TABLE> tag.

Graphics and Images

Graphics and images make any user experience more appealing and should be used, but sparingly, when they add value to the user.

If you wish to display an image that is larger than the working area, Pocket Internet Explorer will follow these rules:

- If "Fit to screen" is OFF, Pocket Internet Explorer displays the image as specified with the <IMG> tags, honoring the "height" and "width" attributes or the original image size, if the attributes are not specified.
- If "Fit to screen" is ON and...
  - The image is smaller then the screen width, the image displays as specified in the <IMG> tag.
  - The image is wider then the screen width, the image shrinks to fit to the screen, but never smaller than one-half of its original width.

Scaling reduces the quality of many images, so you might consider avoiding images wider than the smallest supported viewing area.

Image Detail
It's good practice to avoid large, detailed images because the scaling operation may obscure critical information. If there is a convenient way to convey the same information without using an image, you are probably better off dispensing with the image altogether. If you must use detailed images, however, you should tailor them especially for mobile devices in order to achieve a predictable result.

Image Color
Pocket Internet Explorer will display color images on color displays. On gray scale or monochrome devices, Pocket Internet Explorer converts the color images to black and white. The original color scheme of an image plays an important role in the way Pocket Internet Explorer renders it on gray scale and monochrome devices. The process works best on images that have a high contrast ratio between colors and that have crisp edges in the details of the picture.

Designing Images
The easiest and most predictable way to use images for mobile devices is to make your own custom, small bitmapped images that take into account the reduced screen size. Design your images with clean lines and simple shapes, since more complicated elements tend to appear ragged. The best practice is to use an image that is deliberately simple, rather than a complex image that looks confusing.

Alt Tags
Users may choose not to load images, so it is extremely important to place meaningful alternate text tags in each of your embedded images. Keep in mind that you are trying to convey the message of the missing picture, not describe it.

Image Maps
Pocket Internet Explorer does support image maps. Remember to keep your images small and simple. Also, your image should convey to the user some indication of its function.

Animated GIFs
Pocket Internet Explorer does not support animated GIFs. The first frame of the animated GIF appears to the user, and the rest of the frames composing the animation are stored on the device, taking up valuable space.

General Guidelines, Tips and Tricks

Here are some general guidelines on how to design for Pocket Internet Explorer.

Remember: Screen size is limited on Pocket PCs

You should design your applications keeping the small screen size in mind. Though both a vertical and a horizontal scroll bar are available, try to fit all your information onto one page.

Resolution of Pocket Internet Explorer Is and Is NOT 240x320
All Pocket PCs have a screen resolution of 240x320. Although the user interface of Pocket PC defines two areas that cannot be used for content in the browser:

- the Menu bar on the bottom of the screen
- the Caption bar on the top of the screen

Both bars take 26 pixels space off the vertical resolution. Therefore, if you design pages for Pocket PC that fit on one page, your page must not exceed 240x268. Once your page exceeds 268 vertical pixels the vertical scroll bar will appear and reduces your screen width to 229 pixels. The user can decide to switch off the address bar giving you additional 23 pixels. Since there is no way for a page to determine the state of the address bar, you should not count on it.

When utilizing the full 240 pixels, the horizontal scroll bar will appear, too, because Pocket Internet Explorer needs to provide a way to reveal the remaining 11 pixels with the vertical scroll bar.

If you design your page in a way that does not require a horizontal scroll bar you gain 11 additional vertical pixels. The following Figure shows the pixel extensions of the Pocket Internet Explorer screen components:

Offline Browsing: Important Scenario for Pocket Internet Explorer
Hyperlinks are not available offline except when the user specifies a link-depth greater than one in a Mobile Favorites. Therefore use remote hyperlinks sparingly and present only the most important information to the user. You should design your pages to work well in disconnected mode. A link depth of n means "download the selected page and any page I can get to in n links from that page."

Pages Cannot Be Changed Dynamically
There is no support for DHTML in Pocket Internet Explorer and therefore all of the content on the page is static. You can use XML and XSL as well as JScript and the document.write method to do some dynamic updating of pages.

More Features are Not Always Better
Implement only those features that are absolutely necessary. More can overload the device unnecessarily.

Adding Space with GIF Files
Sometimes you will find that you cannot get the amount of space you want between elements on your page. If this concerns you, you can use a transparent image to wedge extra space between them. Simply build a 1x1 pixel transparent GIF and embed it into the document within an <IMG> tag. Use the "width" and "height" attributes to give it the required dimensions.

One reason to use a "spacer" GIF is to force a line break after a heading. If you have, for example, an article headline immediately followed by an attribution line, you may not want the extra space that heading tags or a <BR> tag adds. You can force a line break and add a couple of pixels of space by simply inserting the same transparent GIF as mentioned above in the text. Also, specify a width of 240 pixels and a height of 1 pixel.

Text in Images
Text in images is a handy way to decorate a page with a title in a font that Pocket Internet Explorer HTML does not support.

If you do choose to place text in an image, you will generally want to use a typeface that was designed specifically for computer screens. There are several excellent one-bit friendly fonts from which to choose. Verdana was designed to render well on bitmapped displays at all resolutions. This naturally makes it quite suitable for use on mobile devices.

But remember, any text you put in bitmaps will not be indexed by any of the current search engines like AltaVista or Yahoo.

Office 2000 as Web Authoring Tool
Of the four Office applications: Access 2000, Excel 2000, PowerPoint 2000, and Word 2000 only PowerPoint 2000 does a browser version check prior to displaying the page. It delivers up a page that tells the user that this may not correctly display in their browser. In most cases the simplest documents generated by these programs will display correctly, however, implementing slightly more complex features will typically generate issues.

In short, it's wisest not use Office 2000 to author pages for Pocket Internet Explorer.

Best Practices for E-Commerce

Personalization Pages

Personalization of content has had a dramatic impact on the way people use the Web. Allowing users to choose the content they wish to include in their page from your Web site will greatly enhance your visitors' experiences. It is likely that you already offer this feature on your Web site. Obvious uses include a portfolio of stocks, a table of cities for basic weather information, etc. Plus, since the offline content is originally stored on the user's desktop, cookies are supported.

For Pocket Internet Explorer personalization to enhance the user experience, add special Pocket Internet Explorer profiles to your offering. But remember, a user might want two profiles: one for the Pocket PC and on for the desktop.

Advertisements

It is now accepted practice to place advertisements on Web pages. For some companies, the sale of such space is the principal revenue source for the Web site. However, because the content may be viewed offline in Pocket Internet Explorer, there is no completely accurate way for a Web site owner to track the number of advertisement impressions made.

Many times banner advertisements are also link to a company's home page. This is usually not helpful to the user because the company's home page is not viewable offline except when the Pocket PC user has opted for a link-depth greater than one during synchronization and "Follow links outside of this page's Web site" is selected. Remember, the important part of your page should be the valuable content for the user, not the advertisement.

Page Header

A Western reader's eye tends to move from the upper left-hand corner of a screen to the lower right. Thus, you should place the most important information such as a heading or company icon in the upper left hand corner. Place all the relevant information and links toward the bottom of the page.

Typically the first two or three lines of a page may include some informative text about the company or content provider. This area can be used very effectively to communicate to the customer regarding the company and further build brand loyalty. This area can also include a small advertisement.

Design Forms with a 240 Pixels Width in Mind

As I mentioned in the HTML tags section above, the maximum width of a Pocket Internet Explorer page should be 240 pixels. Form elements like <INPUT TYPE="TEXT"> or <INPUT TYPE="BUTTON"> are not shrunken by the Fit to window option of Pocket Internet Explorer and will never be rendered wider than the width of the screen. For your e-commerce customers it is especially annoying if they have to scroll horizontally to enter their credit card information or deal with your shopping cart.

Designing E-commerce Forms

Always have the 240-pixel width in mind when you are designing forms. Place the input controls in separate lines instead of going horizontally. Limit your fields to the small screen area. If you want to show a picture of the product selected by the customer, choose to put the description below the picture not next to it.

Best Practices for Information Sites

Beside the best practices for e-commerce sites there are some additional points you should think about as you are designing your information site:

Information sites are perfect for Offline Browsing

Avoid everything that is not supported offline, like form fields, large bitmaps, animated GIFs or excessive advertisements. Many news sites or information sites ask for a short feedback on a given article. Since this requires a post back to the server an offline viewer will not get the same experience. You can add a mailto link on the page since Pocket PC supports sending e-mail messages offline.

Concentrate on the essential content

Do not show any links that Pocket Internet Explorer user are not likely to view anyway like videos, audio links or large picture slide shows.

Use Table ofContent Pages Over Continuously-Linked Pages
Many news and information sites create short pages of the articles and point to the continuation of the article with a "click here to read more..." link at the bottom of the page. This practice is not very useful for Pocket Internet Explorer in an offline scenario. Most users do not change their link-depth in the Mobile Favorites very often. In fact they keep the default (zero). Later on the road they try to click on such a link to read on the article but get a "page not available" error. Even if he or she would like to read the full article it is very hard for some users to figure out what link-depth they actually should use.

To avoid this scenario create a short introductory page containing the links to all pages of the article. This still requires the user to change to a link-depth of "one" but makes it more obvious for him. You can even add a hint on that page, telling the user what link-depth he should enter in their synchronization options.

Scripting

With the increasing importance of rich Web applications, client-side scripting is becoming more and more popular.

Pocket Internet Explorer supports client side JavaScript 1.1 (ECMA-262). VBScript is not supported. As a general rule of thumb, Pocket Internet Explorer supports the Internet Explorer 3.02 DOM (Document Object Model).

There are some issues with scripting in Pocket Internet Explorer that I want to point out:

Scripting Errors are Off by Default

Syntax errors in JScript, missing objects, or other causes of JScript errors are ignored on Pocket Internet Explorer. The script terminates without a message. Switch on the error messages by adding the following registry key:

```
[HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main]
"ShowScriptErrors"=dword:00000001
```

JScript in Pocket Internet Explorer is Not Case Sensitive

JScript in Internet Explorer versioins 4.0 and 5.0 is case sensitive. You may find yourself wondering why one script runs fine on Pocket Internet Explorer but causes errors with Internet Explorer on the desktop, the script might be written accessing object properties, methods, events or collections with the incorrect case. Authors are strongly encouraged to use the correct case when writing any JScripting to ensure compatibility with a wider range of browsers, and to facilitate debugging.

No window.open Function

Pocket Internet Explorer does not support multiple windows. Calls to window.open will fail silently (if script errors are off). Attempts to use <A HREF="..." TARGET="_new"(or any other non-existent frame)> will not invoke a new browser window.

Find the Complete Document Object Model Specification

You can download the very detailed Pocket Internet Explorer DOM specification at:
http://www.microsoft.com/mobile/developer/default.asp
For documentation on the JScript engine that's included in Pocket Internet Explorer, see the JScript documentation at: http://msdn.microsoft.com/scripting/
There's also a good chart about what's not supported at:

Pocket Internet Explorer Scripting Engine Has Same Capabilities as JScript Engine 3.0
For more details on the capabilities of the scripting engine check out the chart at:
http://msdn.microsoft.com/scripting/default.htm?/scripting/jscript/doc/jsoriversioninformation.htm

<OBJECT> Tag Limitations

Pocket Internet Explorer supports Active X controls and allows the methods and properties to be scripted. However there are a few limitations compared to the desktop implementation:

Controls Cannot Be Installed On-the-Fly
Internet Explorer for the desktop supports downloading and installing ActiveX controls. This is not supported on Pocket PC. However, ActiveX controls that already reside on the Pocket PC can be referenced with the <OBJECT> tag in pages.

ActiveX Controls Not Affected by Fit to Screen Option
Even if Fit to screen is selected and the ActiveX control is resized to fit into the screen area, the content of the control will not be resized. If you are creating ActiveX controls to be used in Web pages in Pocket Internet Explorer, design the client area no wider than 240 pixels.

Java Applets Are Not Supported

There is no Java Virtual Machine on Pocket PC, therefore Java applets are not supported.

Security

Pocket Internet Explorer supports all common security schemes:

- SSL 2.0, SSL 3.0 and Server Gate Cryptography (SGC).
- By default Pocket Internet Explorer supports 40-bit security encryption.A 128-bit enhancement pack is available for download at: http://www.microsoft.com/pocketpc/downloads/ssl128.asp
- NTLM authentication as well as clear text authentication.

XML Support

One of the newest components of Pocket Internet Explorer is the MSXML.DLL, which is an Internet Explorer 5.0 component available on Pocket PC.

This component enables the Pocket Internet Explorer to display XML in the standard XML syntax colored view, which you may know from thedesktop. It also allows you to use XSL (Extensible Stylesheet Language) to display XML data in a more user friendly way.

Differences Between MSXML in Microsoft Internet Explorer 5

There are few differences between the version of Microsoft XML for Pocket PC and that exposed in Microsoft Internet Explorer 5. However, there are some features that are not supported on Pocket PC.

No backward compatibility support for the Internet Explorer4 MSXML DOM.
MSXML for Microsoft Internet Explorer 5 includes functionality which mimics the XML DOM exposed in Microsoft Internet Explorer 4. To reduce the memory requirements and ROM size MSXML for Pocket PC does not include this functionality.

No support for Data Binding.
Neither Pocket Internet Explorer, nor MSXML for Pocket PC support Data Binding. Support for Data Binding requires a richer base of HTML layout support.

Watch out for CSS attributes.
Remember, Pocket Internet Explorer does not support cascading style sheets (CSS). Look closely at the XSL transforms you create and make sure that they do not include CSS elements or attributes. Most notably, watch for the use of the STYLE attribute on any HTML tag.

Interfaces of the XML Parser

You can also use the Microsoft XML Parser for your C++ or Visual Basic applications.

How to Use the XMLDOM in eMbedded Visual C++
To access the XMLDOM within an eMbedded Visual C++ application you can include the <MSXML.H> file and add the following lines to the header of your source code:

```
#include <objsafe.h>
namespace MSXML
{
#include <msxml.h>
}
#include <ocidl.h>
```

Here is a small example for a XML Code:

```
MSXML::IXMLDOMDocument          *iXMLDoc     = NULL;
MSXML::IXMLDOMParseError                *pParsingErr = NULL;
MSXML::IXMLDOMElement           *iXMLElm     = NULL;
MSXML::IXMLDOMNodeList          *iXMLChild   = NULL;
MSXML::IXMLDOMNode                      *iXMLItem    = NULL;
HRESULT  hr;
short tEmpty;
BSTR bStr;

hr = CoInitializeEx(NULL,COINIT_MULTITHREADED);
if(!SUCCEEDED(hr))
        return 0;
hr = CoCreateInstance (MSXML::CLSID_DOMDocument, NULL,
        CLSCTX_INPROC_SERVER | CLSCTX_LOCAL_SERVER,
        MSXML::IID_IXMLDOMDocument, (LPVOID *)&iXMLDoc);
if(iXMLDoc)
{
   iXMLDoc->put_async(VARIANT_FALSE);

   // Pocket PC workaround:
   // Remove document safety options
   IObjectSafety *pSafety;
   DWORD dwSupported, dwEnabled;

   if ( SUCCEEDED(iXMLDoc->QueryInterface(
                  IID_IObjectSafety, (void**)&pSafety)))
   {
      pSafety->GetInterfaceSafetyOptions(
               MSXML::IID_IXMLDOMDocument, &dwSupported, &dwEnabled );
      pSafety->SetInterfaceSafetyOptions(
               MSXML::IID_IXMLDOMDocument, dwSupported, 0 );
   }

   iXMLDoc->loadXML(L"<customer><first_name>Joe</first_name>"
            L"<last_name>Smith</last_name></customer>",
           &tEmpty);
   iXMLDoc->get_documentElement(&iXMLElm);
   iXMLElm->get_childNodes(&iXMLChild);
   iXMLChild->get_item(1,&iXMLItem);
   iXMLItem->get_xml(&bStr);
   MessageBox(NULL,bStr,TEXT("Caption"),MB_OK);
}
```

Please take a special look at the "Pocket PC workaround". This three lines are necessary for the MSXML.DLL to Load the XML source. If you do not add those lines, I found that the XMLDLL even tends to lockup.

How to use the XMLDOM in eMbedded Visual Basic
Using the XMLDOM in Visual Basic is just as easy. Here is the same little code snippet showing how to parse XML in eMbedded Visual Basic:

```
Dim xmlDoc
Dim currNode
Dim xml
Set xmlDoc = CreateObject("microsoft.xmldom")
xml = "<customer><first_name>Joe</first_name>"
xml = xml & "<last_name>Smith</last_name></customer>"
xmlDoc.loadXML (xml)
Set currNode = xmlDoc.documentElement.childNodes.item(1)
MsgBox currNode.xml
```

The message box will show:

```
<last_name>Smith</last_name>
```

How to Use the XMLDOM in JavaScript
You can even access the XMLDOM using client-side JavaScript 1.1. Here is the same code using JScript:

```
<XML ID="Contacts">
<CONTACTS>
    <CONTACT>
        <NAME>Stephanie Smith</NAME>
        <BIRTHDATE>1971-07-01</BIRTHDATE>
        <EMAIL>ssmith@abcdef.com</EMAIL>
        <PHONE>(425) 111-1111</PHONE>
    </CONTACT>
    <CONTACT>
        <NAME> Bill Williams</NAME>
        <BIRTHDATE>1968-09-17</BIRTHDATE>
        <EMAIL>billw@abcdef.com</EMAIL>
        <PHONE>(425) 111-1111</PHONE>
    </CONTACT>
    <CONTACT>
        <NAME>Christopher Jones</NAME>
        <BIRTHDATE>1999-09-08</BIRTHDATE>
        <EMAIL>cjones@abcdef.com</EMAIL>
        <PHONE>(425) 111-1111</PHONE>
    </CONTACT>
</CONTACTS>
</XML>

function showPhone()
{
    var root = Contacts.documentElement;
    var selectedElems =
      root.selectNodes("CONTACT[NAME='Bill Williams']");
    var billElem = selectedElems.item(0);
    var phone =
      billElem.childNodes.item(3).nodeTypedValue;

    alert("Bill Williams phone number is " + phone);
}
```

Summary

Pocket Internet Explorer is a powerful Web browser capable of nearly all tasks the modern Web has to offer. If you spend a little time while designing your pages to optimize it for Pocket Internet Explorer you may address a whole now customer base as your Internet audience.

XML is gaining more and more importance on the Web. The XML support found in Pocket Internet Explorer is definitively one of its biggest features. Together with client side JScript you can create powerful Web applications that take full advantage of the mobile Pocket PC.

For More Information

For the latest information on Pocket PC:

http://www.microsoft.com/mobile/pocketpc/

Microsoft Windows CE .NET DOM Property

http://msdn.microsoft.com/library/en-us/wcesoap/htm/cerefDOMProperty.asp

More information on scripting:

http://msdn.microsoft.com/scripting/
http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/
ttp://helpmaster.com/htmlhelp/javascript/popecma262.htm
http://www.mountaindragon.com/javascript/resources.htm

More information about XML:

http://msdn.microsoft.com/library/default.asp?url=/nhp/default.asp?contentid=28000438

Welcome to the MSDN Library

MSDN Home > MSDN Library > Mobile and Embedded Development > Windows Mobile >

Make Your Web Applications Support Pocket PC

Even if Pocket PC has a real Web browser—the Microsoft Internet Explorer for the Pocket PC—the content can look even better with small adjustments. We will look into some of the limitations of the Pocket PC browser to make suitable adjustments to already existing dynamic Web pages.

Applies to:
Microsoft® Windows® Powered Pocket PC 2000
Microsoft Pocket Internet Explorer
Your current Web application toolset
Internet Information Services (IIS)
Microsoft ActiveX® Server Pages (ASP) included in any server version of Microsoft Windows® since Microsoft Windows NT® 4.0
Microsoft Visual Studio®
Download 620-CF.exe

Gotchas

If you want to support multiple clients, be sure to test your client-sensitive content on all clients that you want to support.

Languages Supported

English

Web Application Platforms

The discussion in this article and accompanying samples are focused on the use of the Microsoft Web server, IIS, and ASP. However, most of the logic that is discussed could be applied to a Web application hosted on any server platform.

Who Is the Client?

In IIS, using ASP, there are ways to find out who is the client requesting a page. This is done by looking at some of the HTTP (Hypertext Transfer Protocol) information related to a request. From ASP you get to this information through the ServerVariables collection on the Request object. One that you could use is called **HTTP_USER_AGENT**, and we get the value like this:

```
UserAgent = Request.ServerVariables("HTTP_USER_AGENT")
```

And as the Pocket PC usually return the string:

```
Mozilla/2.0 (compatible; MSIE 3.02; Windows CE; 240x320)
```

We could get a variable indicating that we have a Pocket PC client doing the request by:

```
IsPocketPC = (InStr(UserAgent, "Windows CE") > 0)
```

And as the Microsoft Mobile Explorer (MME) returns something like this:

```
Mozilla/1.22 (compatible; MMEF20; Cellphone; Generic Small)
```

(The "Generic Small" string is replaced by whichever phone MME is running on. An example is the Sony phone that reports "Sony CMD-Z5.") We could get a variable indicating that we have a MME client doing the request by:

```
IsMME = (InStr(UserAgent, "MME") > 0)
```

In the sample code (download 620-CF.exe), you find a testclient.asp file that you can put up in a virtual directory in your IIS (that allow script execution) and browse to the page from different clients to find out what values they generate.

For a more complete discussion on determining clients, please see the white paper Designing Web Sites for the Internet Explorer for Pocket PC.

Transforming Content

Let us start with a fairly common Web application page—an order entry form. This is the ASP page source (only HTML [Hypertext Markup Language] so far and somewhat simplified compared to the samples):

```
<HTML><HEAD><TITLE>New Purchase Order</TITLE></HEAD>
<BODY><FONT FACE="Tahoma">

<!-- LOGO -->
<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0" WIDTH="100%">
  <TR>
    <TD VALIGN="top" WIDTH="100%"><IMG SRC="line.gif"
      WIDTH="100%" HEIGHT="42"></TD>
    <TD VALIGN="top" WIDTH="93"><IMG SRC="logo.gif"
      WIDTH="93" HEIGHT="42"></TD>
  </TR>
</TABLE>
<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0"
  WIDTH="100%"><TR><TD VALIGN="top" WIDTH="100">
        <!-- MENU -->
<B>Menu:</B><BR>
<A HREF="">Home</A><BR>
<A HREF="">About Us</A><BR>
<A HREF="">Order</A>
</TD><TD VALIGN="top">
<H1>New Purchase Order</H1>
<P><B>
  Fill in the information and create new Purchase Order.
</B></P>

<!-- MAIN -->
<H2>Purchase Order Information</H2>
<FORM ACTION="submit.asp" METHOD="POST" id=form1 name=form1>
<TABLE BORDER="0" CELLSPACING="1" CELLPADDING="3">
  <TR>
    <TD VALIGN="top"><B>Customer:</B></TD>
    <TD VALIGN="top"><INPUT TYPE="text" NAME="Customer" VALUE=""></TD>
  </TR>
  <TR>
    <TD VALIGN="top"><B>Order No:</B></TD>
    <TD VALIGN="top"><INPUT TYPE="text" NAME="OrderNo" VALUE=""></TD>
  </TR>
  <TR>
    <TD VALIGN="top"><B>Our Ref:</B></TD>
    <TD VALIGN="top"><INPUT TYPE="text" NAME="OurRef" VALUE=""></TD>
  </TR>
  <TR>
    <TD VALIGN="top"><B>Customer Ref:</B></TD>
    <TD VALIGN="top"><INPUT TYPE="text" NAME="CustRef" VALUE=""></TD>
  </TR>
  <TR>
    <TD VALIGN="top" COLSPAN="2"><INPUT TYPE="submit"
                                  VALUE="Create" id=submit1 name=submit1></TD>
  </TR>
</TABLE>
</FORM>
</TD></TR></TABLE>

</FONT></BODY></HTML>
```

And the page looks like this in Microsoft Internet Explorer 5.5.

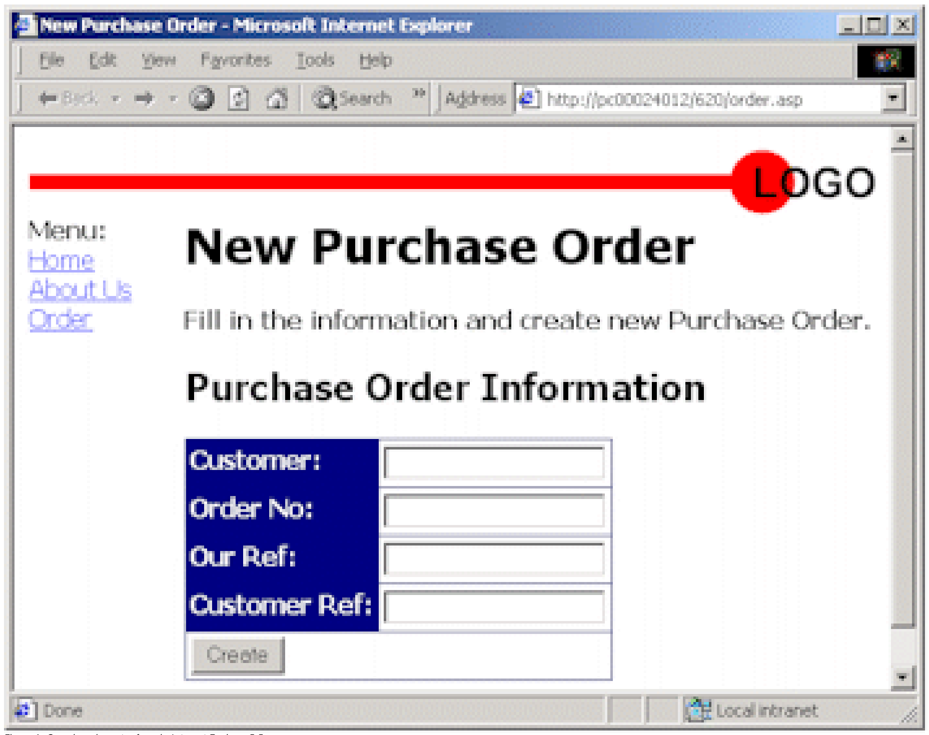

New Purchase Order - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Address http://pc00024012/620/order.asp
LOGO
Menu:
Home
About Us
Order
New Purchase Order
Fill in the information and create new Purchase Order.
Purchase Order Information
Customer:
Order No:
Our Ref:
Customer Ref:
Create
Done
Local intranet

Figure 1: Sample order entry form in Internet Explorer 5.5.

If you look at this page in Internet Explorer for the Pocket PC, you will see that even with the "Fit to Page" option enabled, you get a horizontal scroll bar. It is not very efficient to navigate horizontally with the stylus, so let us see what we can do about that. The menu is placed in a very common way for a normal Web application—in a separate column (created by a table) on the left. In a Pocket PC, we would like to place the menu as a row instead. This saves valuable horizontal space and converts it to compact vertical space instead. We start by adding code to get information about the client:

```
' Device Indicators
UserAgent = Request.ServerVariables("HTTP_USER_AGENT")
IsPocketPC = (InStr(UserAgent, "Windows CE") > 0)
IsMME = (InStr(UserAgent, "MME") > 0)
IsThin = (IsPocketPC Or IsMME)
```

With those variables available, we could modify the menu code to be something like this:

```
<% If Not IsThin Then %><B>Menu:</B><BR><% End If %>
<A HREF="">Home</A><% If IsThin Then %> <% Else %><BR><% End If %<
<A HREF="">About Us</A><% If IsThin Then %> <%Else%><BR><%End If%>
<A HREF="">Order</A>
```

In the same way we could remove the table that creates the extra column for the menu. Also, the first logo row could be changed to a simple logo on a Pocket PC client by:

```
<% If Not IsThin Then %>
  <TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0" WIDTH="100%">
    <TR>
      <TD VALIGN="top" WIDTH="100%"><IMG
        SRC="line.gif" WIDTH="100%" HEIGHT="42"></TD>
      <TD VALIGN="top" WIDTH="93"><IMG
        SRC="logo.gif" WIDTH="93" HEIGHT="42"></TD>
    </TR>
  </TABLE>
<% ElseIf IsPocketPC Then %>
  <IMG SRC="logo.gif" WIDTH="93" HEIGHT="42"><BR>
<% End If %>
```

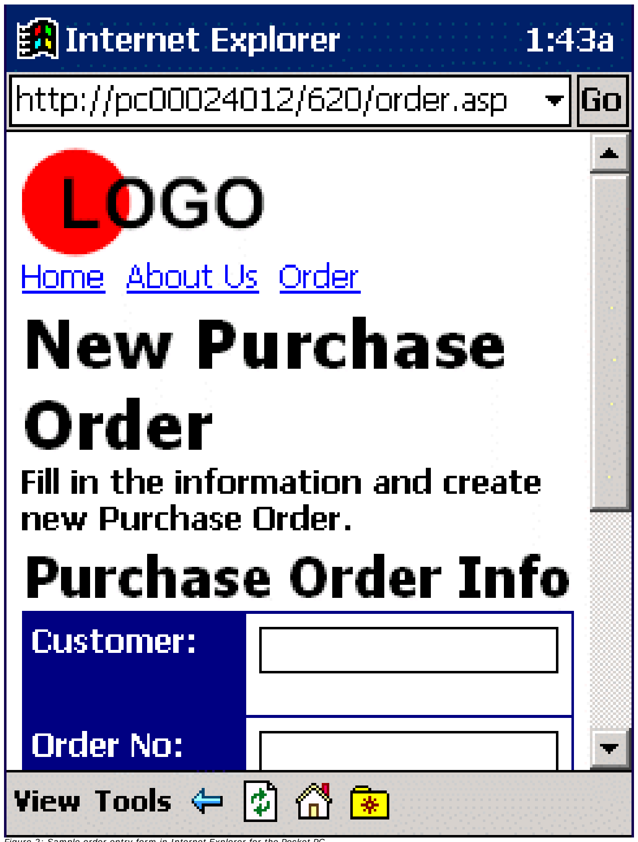
You could even do abbreviations for certain clients by:

```
<H2>Purchase Order Info<%If Not IsThin Then%>rmation<%End If%></H2>
```

An interesting case is when you want to enter information. If you leave **INPUT** fields without any **SIZE** attribute, you will most probably get a field that is wider than the screen, resulting in a horizontal scroll bar again. Therefore it is a good suggestion to include a **SIZE** attribute, like this:

```
<INPUT TYPE="text" NAME="Customer" VALUE=""<% If IsThin Then Response.Write(" SIZE=""10""") %>>/PRE>
```

Let us look at how this looks in the Pocket PC now.



New Purchase Order

Fill in the information and create new Purchase Order.

Purchase Order Info

Customer:	
Order No:	

Figure 2: Sample order entry form in Internet Explorer for the Pocket PC.

And in the samples you will see that I have even added support for MME, and here are some screen shots of the same order form in the MME emulator.

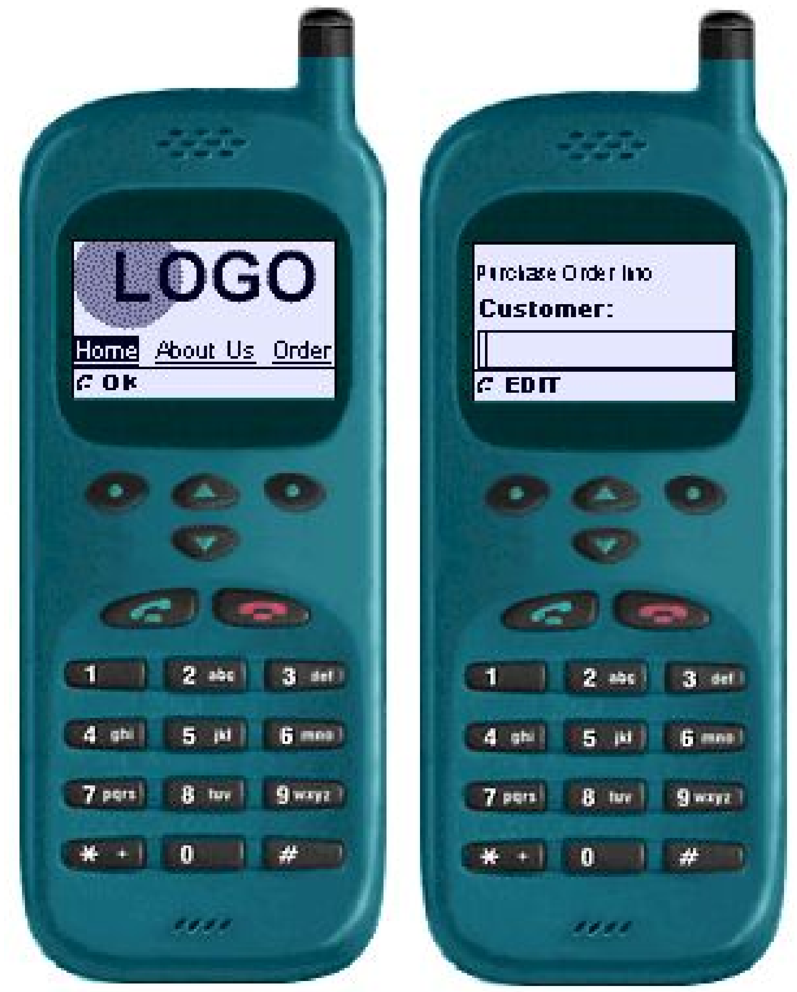


Figure 3: Sample order entry form in MME.

For a complete example, please see the ordermulti.asp file in the sample code (download 620-CF.exe).

More Structure

The above example showed that you could easily update your existing pages with client-specific content. However, a large amount of your content is not real content, and that content you would probably want to put in one place to reuse on many pages. That place is called "server-side includes." For example, the logo and the menu in the above example would be perfect to put in a separate file (top.asp) and then include that file in all pages that need the standard header (logo + menu). Here is the code to do that:

```
<!--#INCLUDE FILE="top.asp"-->
```

You would probably do the same for the code to determine the client. If you do that, and you have a new client that should have the same content as the other "thin" clients (variable IsThin in the above example), you can just add that client type to the include page that determines the client type and all pages that use it will create "thin" content for the new client.

For a complete example, please see the files order.asp, global.asp, top.asp, and bottom.asp in the sample code (download 620-CF.exe).

Even More Structure

If you want to follow this logic all the way, you should start looking in the XML direction. With XML and Extensible Stylesheet Language (XSL) you can separate content from presentation (layout). You can have content in XML and use different XSL style sheets for different clients. You could do this with custom code in ASP or you could use ready-made software for this purpose from Microsoft called XSL ISAPI (Internet Server Application Programming Interface) Filter.

Conclusion

Much of your existing Web applications can be modified to better support the Internet Explorer in Pocket PC. As you have seen, there is not a need for complete rewrite, but rather a "simplification" of existing content.

Why don't you start modifying one of your existing Web applications to better support Pocket PCs and other thin clients today? Just do it a

Is That a Script in Your Pocket?

Andrew Clinick
Microsoft Corporation

August 14, 2000

Download Script0800.exe.

Contents

The great thing about working in the Windows Script group is that you get to see your product used in so many applications, both internally at Microsoft and externally in the hundreds (nay, thousands) of applications using script written by people like you. Traditionally, script has been used in Web applications that run mostly on PCs (be they Windows-, Unix-, or Mac-based)—but now people are starting to develop Web applications for smaller form factors, ranging from WAP-based cell phones to the new Pocket PCs running on Windows CE. I recently received the latest in the Pocket PC range, a Compaq iPaq. I was quite taken by my new Compaq iPaq (it's shiny and silver, and it has an on/off button), so I thought I'd look at how you can use your existing script knowledge to develop applications, both local and Web-based, on Pocket PCs.

All Pocket PCs run Windows CE 3.0, a version of Windows designed to be as compact and modular as possible. As a result, not all the features you've come to expect on a Windows PC are present—but a surprising number are available, including the Windows Script engines. JScript and Visual Basic Scripting Edition (VBScript) are available on all Pocket PCs to provide access to the script engines in applications such as Pocket Internet Explorer and perhaps your next application.

Which Languages You Can Use and Where

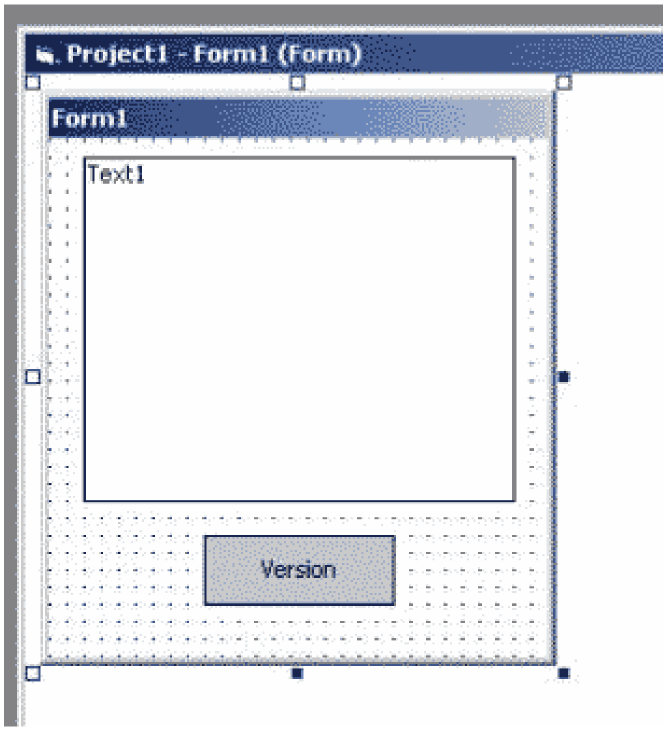
If you're going to develop an application for the Pocket PC, it's important to know what is available to you. Pocket PC ships with a version of Internet Explorer, affectionately known as PIE (Pocket Internet Explorer). PIE has a bunch of interesting features, one of which is the ability to run script code in an HTML page. Because memory is at a premium on Pocket PCs, PIE allows only one script language, JScript, to be used in HTML pages. This provides the best coverage in terms of Web content already out there, and keeping just one language reduces the amount of memory PIE requires when loading a Web page. This means that you can't use VBScript in PIE, but the decision to conserve memory is one you'll probably appreciate when you're running code on a Pocket PC.

The JScript version that runs on Pocket PC is based on the JScript 3.0 code base. The Pocket PC team took the sources from the Windows Script team and looked at how they could reduce the size of the .dll file that implements JScript. The Pocket PC version of JScript is 347 KB, compared with 541 KB for the current 5.5 desktop version. To cut down on size, the Pocket PC version of JScript doesn't have some features, such as regular expressions and VBArray. Apart from that, it's pretty much JScript 3.0. For more information on what's in JScript 3.0, check out the Version Information table. This page tells you exactly which feature was introduced and in which version.

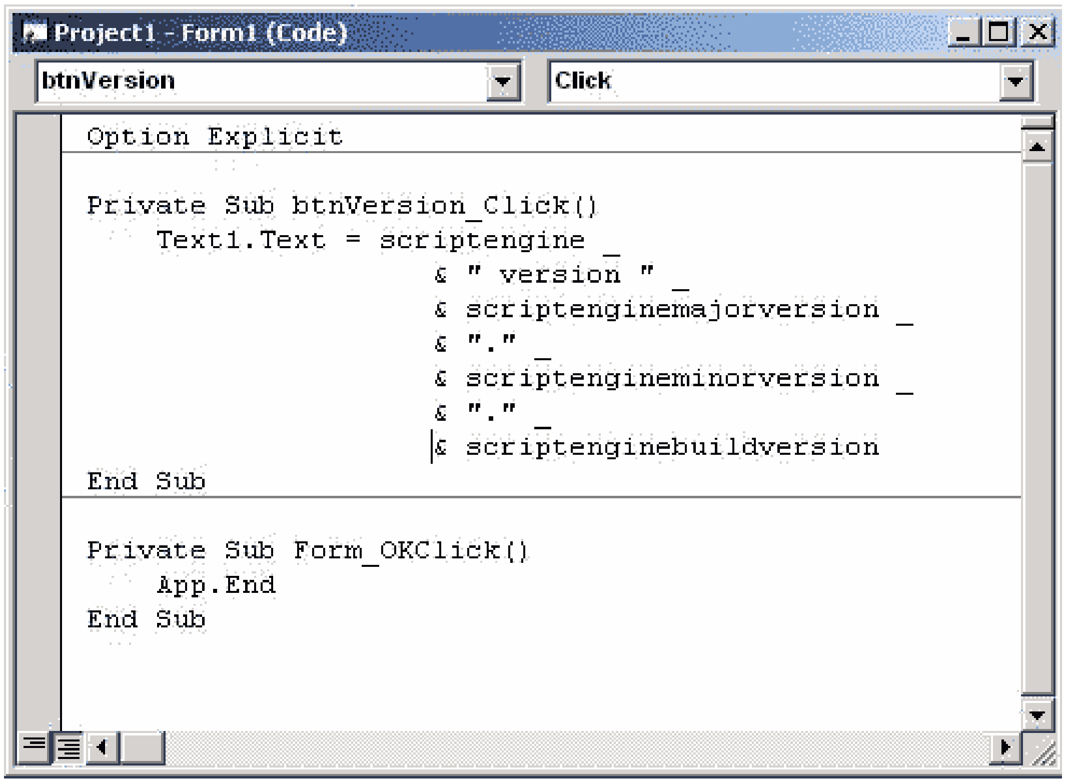
Although PIE doesn't support VBScript, that doesn't mean that VBScript isn't on the Pocket PC platform. VBScript forms the basis of the Visual Basic runtime that is used when you run eMbedded Visual Basic programs. The folks in the embedded tools division took the VBScript source code from the script group and extended it to allow the features required in eMbedded Visual Basic. Like JScript, VBScript is based on the Version 3.0 code base—so it doesn't have the new classes feature added in VBScript 5.0. I highly recommend eMbedded Visual Basic, and best of all, you can order it for just the price of packaging from the Windows CE Web site.

To illustrate how to use the script engines on a Pocket PC, I wrote a simple HTML page and eMbedded Visual Basic application that display the versions of the script engines. If you're a script developer, you're probably aware of the **scriptengine**, **scriptenginemajorversion**, and **scriptengineminorversion** functions in VBScript and JScript. These are useful functions in themselves, because they allow you to check the version of the script engine that your users are running. If it's an old version, you can redirect users to an update.

Using eMbedded Visual Basic, you can create a simple project with a single form, and put a text box and button on the form.

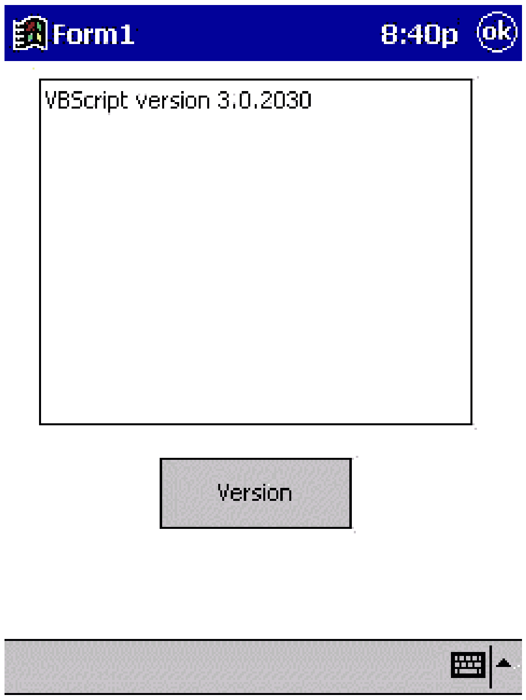


For the **onclick** event handler on the version button, you can add one line of Visual Basic code.



```
Option Explicit

Private Sub btnVersion_Click()
    Text1.Text = scriptengine _
                    & " version " _
                    & scriptenginemajorversion _
                    & "." _
                    & scriptengineminorversion _
                    & "." _
                    & scriptenginebuildversion
End Sub

Private Sub Form_OKClick()
    App.End
End Sub
```

Once this is done, just run the code. To get the application running on your Pocket PC (or the supplied Windows-based emulation), eMbedded Visual Basic does some magic, and up comes the Visual Basic application on your Pocket PC. Click the button, and the relevant version information about VBScript is shown in the text box.



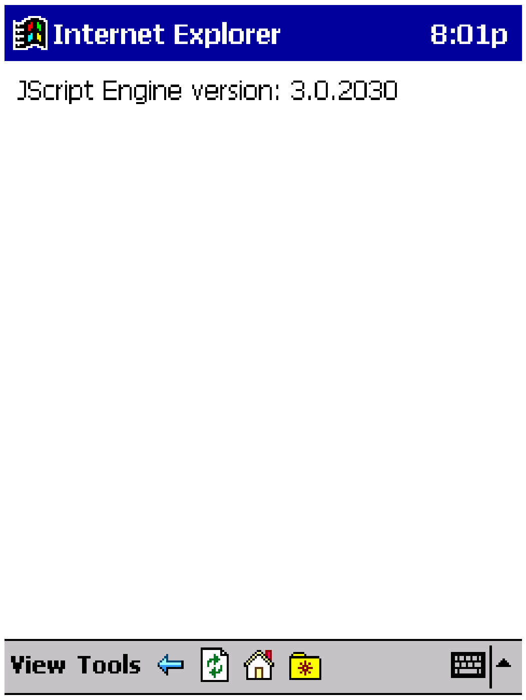
VBScript version 3.0.2030

It's pretty simple to accomplish the same thing in JScript and HTML. Just create an HTML page with a text area and a button object, and hook up some JScript code to the **onclick** event of the button. You do this on a Pocket PC exactly the same way you would on a normal HTML page.

```
<html>
<body>

<script language="JScript">
function getVersion()
{
txtVersion.value = ScriptEngine() + " version " +
ScriptEngineMajorVersion() + "." + ScriptEngineMinorVersion() + "." +
ScriptEngineBuildVersion()
}
</script>

<p align="center"><textarea rows="7" cols="33" name="txtVersion"></textarea></p>
<p align="center"><input type="button"
value="Version" name="btnVersion" onclick="getVersion()"></p>

</body>
</html>
```

Running the HTML on a Pocket PC is not as straightforward as it is on a PC, because it's likely that your Pocket PC isn't connected to a network. PIE does allow direct connection to the Internet via a modem, by using infrared to a cell phone, or by plugging in a compact flash network card. So if you're lucky enough to be able to do that, you can just type in the address of the page, and PIE will load the page.

PIE also provides a great way to deal with offline pages via Mobile Favorites. To use Mobile Favorites, navigate to a page on your Windows Internet Explorer, then choose **Create Mobile Favorite** from the **Tools** menu. (All this assumes you have ActiveSync 3.1 installed.) As soon as you add the page as a Mobile Favorite, ActiveSync will synchronize the page to your Pocket PC. Once the page is downloaded onto your Pocket PC, you can access it via PIE's Favorites (or Favourites) menu. Here's what the JScript version page looks like in PIE.



JScript Engine version: 3.0.2030

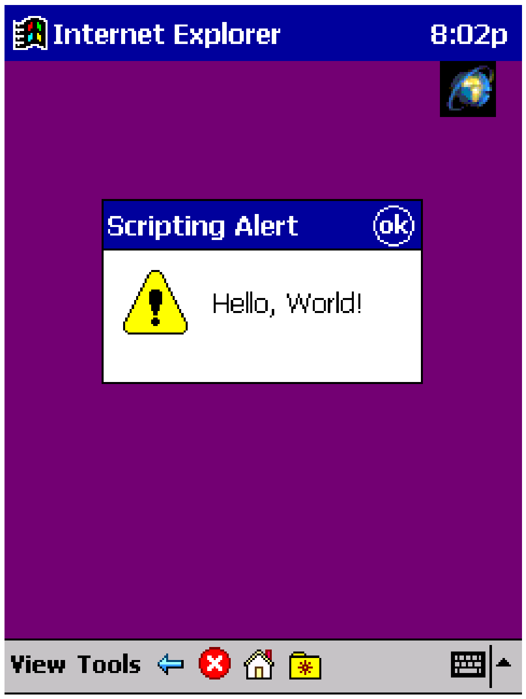
Using Script to Build Applications

To illustrate how you might use script to build applications on Pocket PCs, I've built some simple applications that use some of the features available to you, and work around the difficulties imposed by working in an offline Web world. One of the key challenges is how to get some form of dynamic content to the user without access to the rich object model that Internet Explorer on Windows provides. This is a little like going back to Internet Explorer 3.0 or HTML 3.2—but with the added challenge of having only a 320 x 240 screen to work with (and to think we used to complain about designing for 640 x 480). PIE has a great feature that tries to shrink the page to fit the screen; this certainly helps, but you still need to consider the smaller screen to provide the best user experience.

Getting dynamic content in an HTML 3.2 world means that you have to rely on some old favorites, in particular, **alert** and **document.write**. Luckily, both methods are supported in PIE, and they work pretty much identically to their counterparts in the desktop version of Internet Explorer. A word of caution, though: PIE is stricter about properly formatting your HTML page, so be sure that you have <HTML> and <BODY> tags in place, or you'll spend many a fruitless hour trying to figure out why your script isn't working. (Not that I ever write HTML pages with just <SCRIPT> tags, of course!)

> **Tip** A Microsoft Word version of the HTML object model reference for PIE is available for download. I hope they'll have an HTML version soon.

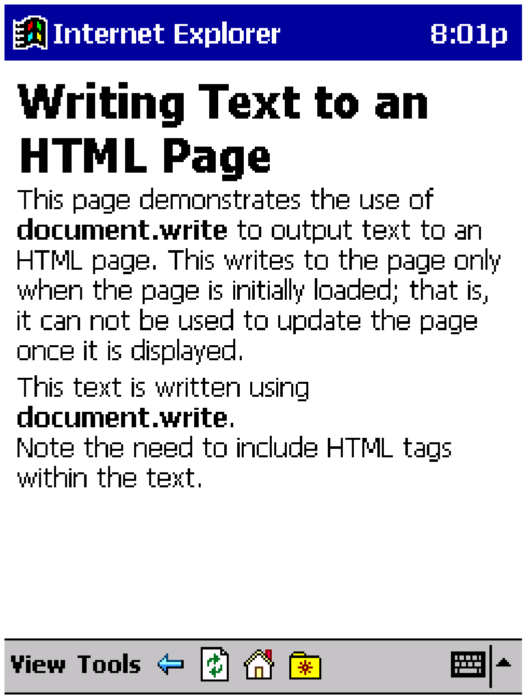
PIE doesn't support the notion of being able to open new windows, so calling **window.open** in your script code won't work. As a result, the only way you can get information to and from the user is via the **alert** and **prompt** methods. Using **alert** results in a slightly different title on the dialog box, and the **OK** button is on the title bar, à la all Pocket PC applications. Apart from those differences, it's pretty much identical to the **alert** method in Internet Explorer.



Hello, World!

The only real way to change the format of a page when it loads is via **document.write**. I'm sure many of you are familiar with **document.write** and its limitations. It does, however, allow you to change the output of a page based on script, and it is as close as you'll get to dynamic content until PIE has full DOM support. Adding DOM support to PIE is not for the faint of heart, especially given the memory footprint of Pocket PCs. I hope Moore's law will come to the rescue, and we'll be able to do all the cool DHTML tricks on Pocket PCs.

```
<HTML>
<BODY>
<H2>Writing Text to an HTML Page</H2>

This page demonstrates the use of <b>document.write</b>
to output text to an HTML page.
This writes to the page only when the page is initially loaded;
that is, it cannot be used to update the page once it is displayed.
<BR>
<SCRIPT LANGUAGE=JScript>
<!-- Start of Script --
document.write ("This text is written using <b>document.write</b>.<BR>");
document.write ("Note the need to include HTML tags within the text.<BR>");
<!-- End of Script -->
</SCRIPT>

</BODY>
</HTML>
```



Writing Text to an HTML Page

This page demonstrates the use of **document.write** to output text to an HTML page. This writes to the page only when the page is initially loaded; that is, it can not be used to update the page once it is displayed.

This text is written using **document.write.**
Note the need to include HTML tags within the text.

I've written a small program to show off some of the possibilities of using script and HTML on a Pocket PC. The application is a simple tip calculator, JScript Tipometer, which lets you work out how much tip you should give and, I hope, solves the eternal problem of how much everyone's share of the check should be. (Side note: As an Englishman, I'm forever confused by how you can pay a check with a check—which came first?) The cool thing about doing this in HTML and script is that my Pocket PC is always running the latest version, because my Mobile Favorite synchronizes whenever I change the page on my Web server. This makes distributing code updates to users a snap; as long as they have their synchronization schedule set up, they'll be up to date.

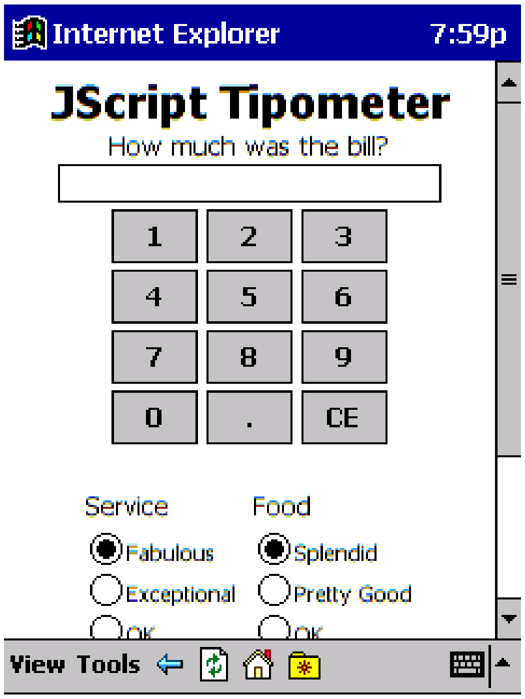
The Tipometer user interface is built with standard HTML form elements. I tried to keep the UI simple, and, where possible, to make it easy for users to enter data without having to use the character recognition software. The character recognition software can be used, but I figured that in a restaurant it would be much easier to use your fingers.

The first thing the Tipometer needs is the amount of the bill. To get this, I added a standard text box and some buttons to ease data entry. The buttons are pretty large, so you can enter data with your finger. Each button has an **onclick** event handler that updates the **value** property of the text box with the value of the button:

```
onclick="txtValue.value += this.value"
```

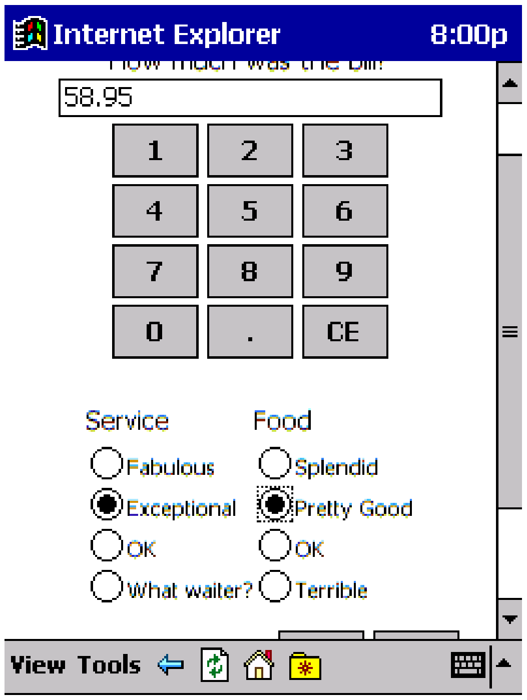
It's pretty simple, but it works. In addition to the numbers, there's a CE button (pardon the pun) that clears the text box.

Once the bill has been entered, the Tipometer asks the user to rate the meal for service and food via a set of radio buttons. By means of a patent-pending tip calculation routine, Tipometer uses the ratings to calculate the tip. This is a pretty low-tech way of working out the tip amount, but you could change it to meet your requirements—and I guess that Tipometer 2.0 would present you with a text box with the calculated tip so you could change it. To align the radio buttons on the smaller screen, I deliberately chose a small font. As you can see, it's clear on screen and pretty readable. To further control the display, I put the radio buttons in a table and centered the table. Putting them in a table seemed to give the best results in this particular case. I recommend that you experiment with what works best for your application.



Once the user has voted on the meal, the only other piece of data required is the number of people among whom the bill will be split. I spent quite a while trying out UI designs for this text box. I finally decided on a simple spinner-control-like UI—because it's unlikely that there are going to be large numbers of people in the party, and I didn't want to take up space with another set of keypad buttons. Because HTML doesn't provide a spinner button, I used a text box (with size set to 2, to minimize space) and two buttons, – and +. Users can click on the buttons the same way they can on a spinner control in Windows, and the **Split between** text box is automatically incremented or decremented. The script to achieve this was pretty trivial—converting the value of the text box to a number and adding or subtracting 1 from it. On the decrementor button, I added a simple check to ensure that you can't reduce the number below 1. Because the number is in a standard text box, the user can also use the character recognizer provided by the operating system, rather than the buttons.

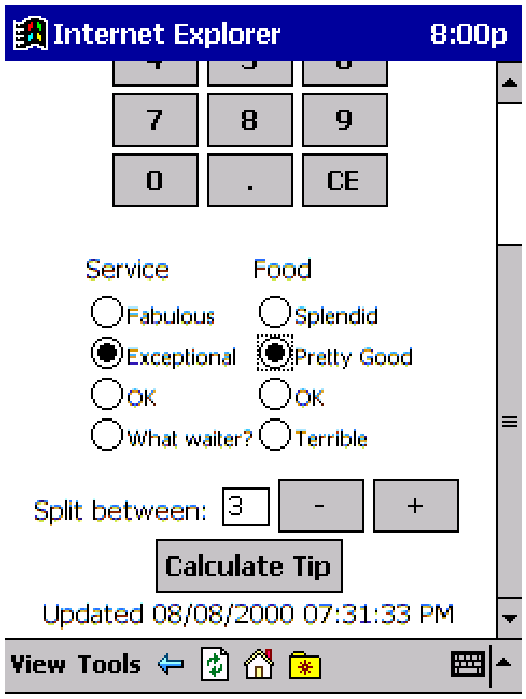
```
  <input type="text" size="2" name="txtParty" value="1">
  <input type="button" value="-" name="btnMinus"
onclick="if (txtParty.value != 1)txtParty.value = Number(txtParty.value) - 1">
  <input type="button" value="+" name="btnPlus"
onclick="txtParty.value = Number(txtParty.value) + 1">
```



Once the data has been collected, all that's left is to calculate the final bill and divide it among the number of people in the party. Nothing is that simple, and trying to put in the arcane rules of tipping and dividing the bill is not a simple task. Because this is a demo application, I took the easy way out and rounded the share of the bill to the nearest dollar. This works most of the time—but if you get an odd number of people in the party, the waiter can be shortchanged by a dollar or so on the tip. It gets worse if you don't leave a tip—but I hope Pocket PC owners are discerning enough to go only to restaurants that offer good service and food. You're free to change the code to meet your needs, and I'd be happy to post updated versions if you send them to msscript@microsoft.com.

```
function btnCalculate_onclick() {
   var foodtip
   var servicetip
   var bill
   var total
   var billshare

   // Get the users view on the food
   for (var i=0;i<Food.length;i++)
   {
   // Get the appropriate value for the food tip
   if (Food[i].checked) foodtip = Number(Food[i].value)
   }

   //  See if the service was any good
   for (var i=0;i<Service.length;i++)
   {
   // Get the appropriate value for the service tip
   if (Service[i].checked) servicetip = Number(Service[i].value)
   }

   // Add the food and service tip together
   var tip = Number(servicetip + foodtip)

   /*   If the tip is less than zero, you shouldn't eat there
   again but you can't give a negative tip, so reset to zero
   */
   if (tip < 0) tip = 0

   bill = Number(txtValue.value)

   tip = Number(bill * (tip/100))

   total = bill + tip

   billshare = total / Number(txtParty.value)


   alert("Total Bill: $" + (Math.round(billshare)*
   Number(txtParty.value)) + "\n" + "Each pay: $" + Math.round(billshare))

}
```

Once the bill and share have been calculated, an alert tells the user what to pay. Initially, I had a text box, but it took up valuable screen real estate, and it wasn't immediately obvious that the result was being shown. An alert seemed to make it obvious.



If you have a Pocket PC, you can try out the JScript Tipometer and add it to your Mobile Favorites.

Hosting Script in Your Applications

A large number of programs that run on Windows host script (VBScript or JScript) in their applications via the Windows Script Interfaces (IActiveScript) and the Windows Script Control. The good news is that the IActiveScript interfaces are fully supported on Windows CE (that's how Pocket IE hooks up JScript), so you can port your existing Windows applications to Windows CE with the embedded tools and take advantage of the script engines that ship with the operating system. At the moment, we don't have the Window Script Control running on Windows CE, but we'd love to hear feedback on how useful it would be for people, so please e-mail us at msscript@microsoft.com. There are already a number of articles (The Visual Programmer, MSJ October 1999 and Say Goodbye to Macro Envy with Active Scripting, for example) about using the Windows Script Interfaces, so I won't go into detail here.

Summary

Pocket PC provides a really rich programming environment that you can take advantage of today. The scripting support provided in the operating system and within Internet Explorer makes it even easier to get up to speed with application development. I hope I've given you an insight into what is possible to build today with your existing scripting skills and a little imagination. I recommend that you take a look at the Pocket PC and embedded tools Web sites for more information about developing applications for Pocket PC.

Andrew Clinick *is a program manager in the Microsoft Script Technology group, so chances are, if there's script involved, he's probably had something to do with it. He spends most of his spare time trying to get decent rugby coverage on U.S. television and explaining cricket to his American colleagues.*

Programmer's Guide to Internet Explorer for Microsoft Windows CE 3.0

Microsoft Corporation

June 2000

Summary: This white paper discusses the differences between the Windows CE-based and the Win32-based versions of Internet Explorer, and the technologies that are supported by Internet Explorer for Windows CE. (14 printed pages)

Contents

Windows CE-based and Win32-based Versions of Internet Explorer

Internet Explorer for Microsoft® Windows CE supports the same features that Microsoft Internet Explorer version 4.0 for Windows-based desktop platforms supports, except for the following features:

- Data binding
- Microsoft Visual Basic®, Scripting Edition (VBScript)
- Extensible Markup Language (XML)
- Downloadable Microsoft ActiveX® controls
- Portable Network Graphics (PNG)
- Microsoft virtual machine
- Gopher
- Recreation Software Advisory Council on the Internet (RSACi) rating system
- Filters and transitions

Microsoft JScript®, which is compatible with the European Computer Manufacturers Association (ECMA) 262 Language Specification, supports the same features that the version of JScript for Windows-based desktop platforms supports, except for the following features:

- **RegExp** support
- SAFEARRAY support, which is used for coexistence with VBScript and other languages
- Scrrun.dll support; including dictionary object, file object, and so on
- Automatic loading of type libraries; in other words, the **IActiveScript::AddTypeLib** method is not supported
- Cross-window object references; for example, opener.top.location

Internet Explorer for Windows CE is not intended to be an upgrade of the Microsoft Pocket Internet Explorer browser, so Internet Explorer for Windows CE does not attempt to provide backwards compatibility with Pocket Internet Explorer or its custom interfaces.

Technologies Supported by Internet Explorer for Windows CE

Internet Explorer for Windows CE supports the following technologies:

- Dynamic HTML (DHTML)
- Windows CE-based Internet functions
- Asynchronous pluggable protocols
- URL monikers
- MSHTML
- Web Browser control
- MLang

Internet Explorer for Windows CE also supports:

- ActiveX controls that are installed on the Windows CE device, but Internet Explorer for Windows CE does not support downloading these controls
- Cascading style sheet (CSS) support, compliant with CSS1 core functionality
- Full event model for all tags and objects
- Integration of JScript and objects
- International language support, including Unicode characters
- Images in GIF, JPEG, BMP, and XBM formats
- Multimedia, using Microsoft DirectShow®

The scripting documentation contained in the MSDN Library (online or CD) includes information on the application programming interfaces (APIs) that Windows CE supports. In this online documentation, you can determine if Windows CE supports an API by moving the mouse pointer over an object name in the **Applies To** lists on the pages for properties, methods, events, and collections. See the **onclick** event in the DHTML Events reference for an example. Later sections of this document also contain information about APIs that Windows CE supports.

Currently, the C, C++, and COM reference pages in the MSDN Library for APIs that Internet Explorer for Windows CE supports include a Windows CE section. This section indicates the version of Windows CE that supports the API and the minimum version of Internet Explorer that is required. See the **InternetOpen** function reference in the Win32 Internet functions reference for an example of a Windows CE section. The C, C++, and COM references that do not have a Windows CE section are not supported currently in Windows CE.

DHTML

DHTML allows you to create Web pages and other documents that automatically and instantly adapt to specific users, user requests, and the changing state of data from sources on the Web and other locations. By using the scripting language of your choice, you can access and manipulate all of the elements of a Web page—tags, attributes, images, objects, and text—to create, move, and modify these elements as needed.

The following table shows the DHTML objects that Internet Explorer for Windows CE supports.

A	**COMMENT**	**HTML**	**NOFRAMES**	**STYLE**
ACRONYM	**DD**	**I**	**NOSCRIPT**	**styleSheet**
ADDRESS	**DEL**	**IFRAME**	**OBJECT**	**SUB**
APPLET	**DFN**	**IMG**	**OL**	**SUP**
AREA	**DIR**	**INPUT**	**OPTION**	**TABLE**
B	**DIV**	**INS**	**P**	**TBODY**
BASE	**DL**	**ISINDEX**	**PARAM**	**TD**
BASEFONT	**document**	**KBD**	**PLAINTEXT**	**TEXTAREA**
BGSOUND	**DT**	**LABEL**	**PRE**	**TextRange**
BIG	**EM**	**LEGEND**	**Q**	**TFOOT**
BLOCKQUOTE	**EMBED**	**LI**	**S**	**TH**
BODY	**event**	**LINK**	**SAMP**	**THEAD**
BR	**FIELDSET**	**LISTING**	**screen**	**TITLE**
BUTTON	**FONT**	**location**	**SCRIPT**	**TR**
CAPTION	**FORM**	**MAP**	**SELECT**	**TT**
CENTER	**FRAME**	**MARQUEE**	**selection**	**U**
CITE	**FRAMESET**	**MENU**	**SMALL**	**UL**
clientInformation	**HEAD**	**META**	**SPAN**	**VAR**
CODE	**history**	**navigator**	**STRIKE**	**WBR**
COL	**Hn**	**NEXTID**	**STRONG**	**window**
COLGROUP	**HR**	**NOBR**	**style**	**XMP**

The following table shows the DHTML methods that Internet Explorer for Windows CE supports. See the MSDN Library to determine the objects for which these methods apply. Windows CE does not always apply these methods to all of the objects to which the methods apply on Windows-based desktop platforms.

add	**createTFoot**	**insertRow**	**remove**
AddDesktopComponent	**createTHead**	**isEqual**	**removeAttribute**

addImport	deleteCaption	IsSubscribed	replace
addRule	deleteCell	item	reset
alert	deleteRow	javaEnabled	resizeBy
assign	deleteTFoot move	resizeTo	
back	deleteTHead	moveBy	scrollBy
blur	duplicate	moveEnd	scrollIntoView
clear	elementFromPoint	moveStart	scrollTo
clearInterval	empty	moveTo	select
clearTimeout	execScript	moveToBookmark	setAttribute
click	expand	moveToPoint	setEndPoint
close	findText	navigate	setInterval
collapse	focus	NavigateAndFind	setTimeout
compareEndPoints	forward	nextPage	showModalDialog
componentFromPoint	getAttribute	open	start
confirm	getBookmark	parentElement	stop
contains	go	pasteHTML	submit
createCaption	ImportExportFavorites	previousPage	tags
createElement	inRange	prompt	taintEnabled
createRange	insertAdjacentHTML	refresh	write
createStyleSheet	insertAdjacentText	reload	writeln
createTextRange	insertCell		

The following table shows the DHTML events that Internet Explorer for Windows CE supports. See the MSDN Library to determine the objects for which these events apply. Windows CE does not always apply these events to all of the objects to which the events apply on Windows-based desktop platforms.

onabort	onerror	onload	onresize
onblur	onfinish	onmousedown	onscroll
onbounce	onfocus	onmousemove	onselect
onchange	onhelp	onmouseout	onselectstart
onclick	onkeydown	onmouseover	onstart
ondblclick	onkeypress	onmouseup	onsubmit
ondragstart	onkeyup	onreset	onunload

The following table shows the DHTML collections that Internet Explorer for Windows CE supports. See the MSDN Library to determine the objects for which these collections apply. Windows CE does not always apply these collections to all of the objects to which the collections apply on Windows-based desktop platforms.

all	elements	options
anchors	embeds	plugins
applets	forms	rows
areas	frames	rules
bookmarks	images	scripts
cells	imports	styleSheets
children	links	tBodies

The following table shows the DHTML properties that Internet Explorer for Windows CE supports. See the MSDN Library to determine the objects for which these properties apply. Windows CE does not always apply these properties to all of the objects to which the properties apply on Windows-based desktop platforms.

!important	colorDepth	leftMargin	readOnly
accessKey	cols	length	readyState
action	colSpan	letterSpacing	referrer
activeElement	compact	lineHeight	rel
align	complete	link	returnValue
aLink	content	linkColor	rev
aLinkColor	cookie	listStyle	right
alt	cookieEnabled	listStyleImage	rightMargin
altHTML	coords	listStylePosition	rowIndex
altKey	cpuClass	listStyleType	rows
appCodeName	ctrlKey	loop	rowSpan
appMinorVersion	cursor	lowsrc	rules
appName	data	margin	scopeName
appVersion	defaultCharset	marginBottom	scroll
availHeight	defaultChecked	marginHeight	scrollAmount
availWidth	defaultSelected	marginLeft	scrollDelay
background	defaultStatus	marginRight	scrollHeight
backgroundAttachment	defaultValue	marginTop	scrolling
backgroundColor	defer	marginWidth	scrollLeft
backgroundImage	dialogArguments	maxLength	scrollTop

backgroundPosition	dialogHeight	media	search
backgroundPositionX	dialogLeft	menuArguments	sectionRowIndex
backgroundPositionY	dialogTop	method	selected
backgroundRepeat	dialogWidth	Methods	selectedIndex
balance	direction	multiple	self
behavior	disabled	name	shape
bgColor	display	noHref	shiftKey
bgProperties	domain	noResize	size
border	encoding	noShade	sourceIndex
borderBottom	event	noWrap	span
borderBottomColor	expando	object	src
borderBottomStyle	face	offsetHeight	srcElement
borderBottomWidth	fgColor	offsetLeft	start
borderColor	fileCreatedDate	offsetParent	status
borderColorDark	fileModifiedDate	offsetTop	styleFloat
borderColorLight	fileSize	offsetWidth	systemLanguage
borderLeft	fileUpdatedDate	offsetX	tabIndex
borderLeftColor	font	offsetY	tableLayout
borderLeftStyle	fontFamily	onLine	tagName
borderLeftWidth	fontSize	opener	tagUrn
borderRight	fontSmoothingEnabled	outerHTML	target
borderRightColor	fontStyle	outerText	text
borderRightStyle	fontVariant	overflow	textAlign
borderRightWidth	fontWeight	overflowX	textDecoration
borderStyle	form	overflowY	textDecorationLineThrough
borderTop	frame	owningElement	textDecorationNone
borderTopColor	frameBorder	padding	textDecorationOverline
borderTopStyle	frameSpacing	paddingBottom	textDecorationUnderline
borderTopWidth	hash	paddingLeft	textIndent
borderWidth	height	paddingRight	textTransform
bottom	hidden	paddingTop	tFoot
bottomMargin	host	pageBreakAfter	tHead
boundingHeight	hostname	pageBreakBefore	title
boundingLeft	href	palette	toElement
boundingTop	hspace	parent	top
boundingWidth	htmlFor	parentElement	topMargin
browserLanguage	htmlText	parentStyleSheet	trueSpeed
bufferDepth	httpEquiv	parentTextEdit	type
cancelBubble	id	parentWindow	units
caption	indeterminate	pathname	updateInterval
cellIndex	index	pixelBottom	URL
cellPadding	innerHTML	pixelHeight	urn
cellSpacing	innerText	pixelLeft	useMap
checked	isMap	pixelRight	userAgent
classid	isTextEdit	pixelTop	userLanguage
className	keyCode	pixelWidth	vAlign
clear	lang	platform	value
clientHeight	language	port	verticalAlign
clientLeft	lastModified	posBottom	visibility
clientTop	layoutGrid	posHeight	vLink
clientWidth	layoutGridChar	position	vlinkColor
clip	layoutGridCharSpacing	posLeft	volume
closed	layoutGridLine	posRight	vspace
code	layoutGridMode	posTop	width
codeBase	layoutGridType	posWidth	wrap
codeType	left	protocol	zIndex
color			

The following table shows the DHTML input types that Internet Explorer for Windows CE supports.

button	file	image	radio	submit
checkbox	hidden	password	reset	text

Windows CE-based Internet Functions

The Windows CE Internet API (WinInet) is an API that is used for Internet client application development. The Wininet.dll module exports **WinInet** functions that are used to develop Internet applications, such as Web browsers and FTP applications.

WinInet is similar to WinInet for Windows-based desktop platforms, with the following exceptions:

- Microsoft ActiveX controls are not supported.
- The Gopher protocol and Gopher functions are not supported.

WinInet enables multithreaded applications to make concurrent calls to **WinInet** functions from different threads. **WinInet** functions synchronize if necessary, but these functions do not validate parameters.

The following table shows the **WinInet** functions that Internet Explorer for Windows CE supports.

InternetCanonicalizeUrl	**InternetQueryDataAvailable**
InternetCloseHandle	**InternetQueryOption**
InternetCombineUrl	**InternetReadFile**
InternetConnect	**InternetReadFileEx**
InternetCrackUrl	**InternetSetCookie**
InternetCreateUrl	**InternetSetFilePointer**
InternetErrorDlg	**InternetSetOption**
InternetFindNextFile	**InternetSetStatusCallback**
InternetGetCookie	**InternetTimeFromSystemTime**
InternetGetLastResponseInfo	**InternetTimeToSystemTime**
InternetLockRequestFile	**InternetUnlockRequestFile**
InternetOpen	**InternetWriteFile**
InternetOpenUrl	

Asynchronous Pluggable Protocols

Asynchronous pluggable protocols enable developers to create pluggable protocol handlers, Multipurpose Internet Mail Extension (MIME) filters, and name-space handlers that work with Internet Explorer for Windows CE and a Uniform Resource Locator (URL) moniker.

Applications can use pluggable protocol handlers to handle a custom URL protocol scheme or filter data for a designated MIME type.

The ability to handle a custom URL protocol scheme by using a pluggable protocol handler allows developers to implement new or custom protocol schemes for Internet Explorer for Windows CE and for applications that use URL monikers. The default pluggable protocol handler that is included with Internet Explorer for Windows CE handles existing protocol schemes, such as Hypertext Transfer Protocol (HTTP) and File Transfer Protocol (FTP).

Pluggable MIME filters can be used to filter data for a particular MIME type. Unlike standard pluggable protocol handlers and pluggable name-space handlers, which only provide data, pluggable MIME filters both read and provide data.

The following table shows the interfaces and methods that are related to asynchronous pluggable protocols that Internet Explorer for Windows CE supports.

IInternetBindInfo	**IInternetProtocolRoot::Continue**
IInternetBindInfo::GetBindInfo	**IInternetProtocolRoot::Resume**
IInternetBindInfo::GetBindString	**IInternetProtocolRoot::Start**
IInternetProtocol	**IInternetProtocolRoot::Terminate**
IInternetProtocol::LockRequest	**IInternetProtocolSink**
IInternetProtocol::Read	**IInternetProtocolSink::ReportData**
IInternetProtocol::Seek	**IInternetProtocolSink::ReportProgress**
IInternetProtocol::UnlockRequest	**IInternetProtocolSink::ReportResult**
IInternetProtocolInfo	**IInternetProtocolSink::Switch**
IInternetProtocolInfo::CombineUrl	**IInternetSession**
IInternetProtocolInfo::CompareUrl	**IInternetSession::RegisterMimeFilter**
IInternetProtocolInfo::ParseUrl	**IInternetSession::RegisterNameSpace**
IInternetProtocolInfo::QueryInfo	**IInternetSession::UnregisterMimeFilter**
IInternetProtocolRoot	**IInternetSession::UnregisterNameSpace**
IInternetProtocolRoot::Abort	

The following table shows the functions that are related to asynchronous pluggable protocols that Internet Explorer for Windows CE supports.

CoInternetGetProtocolFlags	**CoInternetGetSession**	**CoInternetParseUrl**

The following table shows the structures that are related to asynchronous pluggable protocols that Internet Explorer for Windows CE supports.

PROTOCOLDATA	**PROTOCOLFILTERDATA**

The following table shows the enumerations that are related to asynchronous pluggable protocols that Internet Explorer for Windows CE supports.

BINDSTRING	**FLAGS**	**PARSEACTION**

URL Monikers

A URL moniker is a system-provided moniker class that supports asynchronous binding to a URL.

Monikers are used as the basis for linking in OLE. After a moniker is bound to an object, the IMoniker interface for the moniker can be used to locate, activate, and access the bound object without having any other specific information on where the actual object is located. For standard monikers, this binding process occurs synchronously, which does not impact performance dramatically, because the moniker and object are usually on a local system. Examples of objects that can be bound to a moniker include files, items, and pointers.

Binding a moniker to a URL synchronously impacts performance, because the process has to wait for responses from the network before completing the binding process. Using asynchronous URL monikers and the URL functions allows an application to bind a moniker to a URL without affecting performance.

The following table shows the interfaces and methods that are related to URL monikers that Internet Explorer for Windows CE supports.

IAuthenticate	**IBindStatusCallback::OnStartBinding**
IAuthenticate::Authenticate	**IBindStatusCallback::OnStopBinding**
IBindHost	**IHttpNegotiate**
IBindHost::CreateMoniker	**IHttpNegotiate::BeginningTransaction**
IBindHost::MonikerBindToObject	**IHttpNegotiate::OnResponse**
IBindHost::MonikerBindToStorage	**IHttpSecurity**
IBinding	**IHttpSecurity::OnSecurityProblem**
IBinding::Abort	**IPersistMoniker**
IBinding::GetBindResult	**IPersistMoniker::GetCurMoniker**
IBinding::GetPriority	**IPersistMoniker::IsDirty**
IBinding::Resume	**IPersistMoniker::Load**
IBinding::SetPriority	**IPersistMoniker::Save**
IBinding::Suspend	**IPersistMoniker::SaveCompleted**
IBindStatusCallback	**IWindowForBindingUI**
IBindStatusCallback::GetBindInfo	**IWindowForBindingUI::GetWindow**
IBindStatusCallback::GetPriority	**IWinInetHttpInfo**
IBindStatusCallback::OnDataAvailable	**IWinInetHttpInfo::QueryInfo**
IBindStatusCallback::OnObjectAvailable	**IWinInetInfo**
IBindStatusCallback::OnProgress	**IWinInetInfo::QueryOption**

The following table shows the functions that are related to URL monikers that Windows CE supports.

CoInternetCombineUrl	**GetClassFileOrMime**
CoInternetCompareUrl	**IsAsyncMoniker**
CoInternetQueryInfo	**ObtainUserAgentString**
CreateAsyncBindCtxEx	**RegisterBindStatusCallback**
CreateURLMoniker	**ReleaseBindInfo**
FindMimeFromData	**RevokeBindStatusCallback**

The following table shows the structures and enumerations that are related to URL monikers that Windows CE supports.

BINDF	**BINDINFOF**	**BINDSTRING**	**BSCF**
BINDINFO	**BINDSTATUS**	**BINDVERB**	**QUERYOPTION**

The following table shows the error values that are related to URL monikers that Windows CE supports.

INET_E_AUTHENTICATION_REQUIRED	INET_E_NO_SESSION
INET_E_CANNOT_CONNECT	INET_E_NO_VALID_MEDIA
INET_E_CANNOT_INSTANTIATE_OBJECT	INET_E_OBJECT_NOT_FOUND
INET_E_CANNOT_LOAD_DATA	INET_E_QUERYOPTION_UNKNOWN
INET_E_CANNOT_LOCK_REQUEST	INET_E_REDIRECT_TO_DIR
INET_E_CANNOT_REPLACE_SFP_FILE	INET_E_REDIRECTING
INET_E_CODE_DOWNLOAD_DECLINED	INET_E_RESOURCE_NOT_FOUND
INET_E_CONNECTION_TIMEOUT	INET_E_RESULT_DISPATCHED

INET_E_DATA_NOT_AVAILABLE	INET_E_SECURITY_PROBLEM
INET_E_DEFAULT_ACTION	INET_E_UNKNOWN_PROTOCOL
INET_E_DOWNLOAD_FAILURE	INET_E_USE_DEFAULT_PROTOCOLHANDLER
INET_E_INVALID_REQUEST	INET_E_USE_DEFAULT_SETTINGS
INET_E_INVALID_URL	

MSHTML

MSHTML is a rendering engine and parser for HTML. Introduced in Microsoft Internet Explorer 4.0, it is the main HTML component of the Internet Explorer Web browser and can be reused in other applications. It hosts ActiveX controls and supports the OLE Control 96 (OC96) specification for windowless controls. Controls that use the OC96 interfaces can gain in performance, and they can be transparent and have an irregular shape.

The following table shows the MSHTML interfaces that Internet Explorer for Windows CE supports. To determine the methods that Windows CE supports for each of these interfaces, see the MSDN Library.

IHTMLAnchorElement	**IHMTLFormElement**	**IHTMLOptionElementFactory**
IHTMLAreaElement	**IHTMLFrameBase**	**IHTMLOptionsHolder**
IHTMLAreasCollection	**IHTMLFrameElement**	**IHTMLParaElement**
IHTMLBaseElement	**IHTMLFramesCollection2**	**IHMTLPluginsCollection**
IHTMLBaseFontElement	**IHTMLFrameSetElement**	**IHTMLRuleStyle**
IHTMLBGsound	**IHTMLHeaderElement**	**IHTMLScreen**
IHTMLBlockElement	**IHTMLHRElement**	**IHTMLScriptElement**
IHTMLBodyElement	**IHTMLIFrameElement**	**IHTMLSelectElement**
IHTMLBRElement	**IHTMLImageLElementFactory**	**IHTMLSelectionElement**
IHTMLButtonElement	**IHTMLImgElement**	**IHTMLSpanFlow**
IHTMLCommentElement	**IHTMLInputButtonElement**	**IHTMLStyle**
IHTMLControlElement	**IHTMLInputFileElement**	**IHTMLStyleElement**
IHTMLControlRange	**IHTMLInputHiddenElement**	**IHTMLStyleSheet**
IHTMLDataBinding	**IHTMLInputImageElement**	**IHTMLStyleSheetRule**
IHTMLDDElement	**IHTMLInputTextElement**	**IHTMLStyleSheetRulesCollection**
IHTMLDialog	**IHTMLIsIndexElement**	**IHTMLStyleSheetsCollection**
IHTMLDivElement	**IHTMLLabelElement**	**IHTMLTable**
IHTMLDivPosition	**IHTMLLegendElement**	**IHTMLTableCaption**
IHTMLDListElement	**IHTMLLIElement**	**IHTMLTableCell**
IHTMLDocument	**IHTMLLinkElement**	**IHTMLTableCol**
IHTMLDocument2	**IHTMLLocation**	**IHTMLTableRow**
IHTMLDTElement	**IHTMLMapElement**	**IHTMLTableSection**
IHTMLElement	**IHTMLMarqueeElement**	**IHTMLTextAreaElement**
IHTMLElementCollection	**IHTMLMetaElement**	**IHTMLTextContainer**
IHTMLEmbedElement	**IHTMLMimeTypesCollection**	**IHTMLTextElement**
IHTMLEventObj	**IHTMLNextIdElement**	**IHTMLTitleElement**
IHTMLFieldSetElement	**IHTMLObjectElement**	**IHTMLTxtRange**
IHTMLFiltersCollection	**IHTMLOListElement**	**IHTMLUListElement**
IHTMLFontElement	**IHTMLOpsProfile**	**IHTMLWindow2**
IHTMLFontNamesCollection	**IHTMLOptionButtonElement**	**IOmHistory**
IHTMLFontSizesCollection	**IHTMLOptionElement**	**IOmNavigator**

The following table shows the MSHTML element events that Internet Explorer for Windows CE supports.

onabort	onerror	onkeydown	onload	onscroll
onblur	onfocus	onkeypress	onmouseup	onselectstart
onclick	onhelp	onkeyup	onresize	

The following table shows the MSHTML document events that Internet Explorer for Windows CE supports.

onclick	onkeydown	onkeyup	onselectstart
onhelp	onkeypress	onmouseup	

Internet Explorer for Windows CE does not support any MSHTML element events2.

Web Browser Control

The Shdocvw.dll component, which is referred to more frequently as the Web Browser control, supplies the functionality that is associated with navigation, in-place linking, management of favorites and history, and support of PICS. In-place linking refers to the ability to click a link in the HTML of the loaded document and load a new HTML document in the same instance of the Web Browser control. This dynamic-link library (or DLL) also exposes interfaces to its host to allow it to be hosted separately as an ActiveX control.

The following table shows the interfaces and methods that are related to the Web Browser control that Windows CE supports.

DWebBrowserEvents2	IWebBrowser2::get_RegisterAsBrowser
DWebBrowserEvents2::BeforeNavigate2	IWebBrowser2::get_Resizable
DWebBrowserEvents2::CommandStateChange	IWebBrowser2::get_Silent
DWebBrowserEvents2::DocumentComplete	IWebBrowser2::get_Top
DWebBrowserEvents2::DownloadBegin	IWebBrowser2::get_TopLevelContainer
DWebBrowserEvents2::DownloadComplete	IWebBrowser2::get_Visible
DWebBrowserEvents2::NavigateComplete2	IWebBrowser2::get_Width
DWebBrowserEvents2::NewWindow2	IWebBrowser2::GetProperty
DWebBrowserEvents2::OnQuit	IWebBrowser2::GoBack
DWebBrowserEvents2::OnVisible	IWebBrowser2::GoForward
DWebBrowserEvents2::ProgressChange	IWebBrowser2::GoHome
DWebBrowserEvents2::PropertyChange	IWebBrowser2::GoSearch
DWebBrowserEvents2::StatusTextChange	IWebBrowser2::Navigate
DWebBrowserEvents2::TitleChange	IWebBrowser2::Navigate2
IWebBrowser2	IWebBrowser2::put_FullScreen
IWebBrowser2::ClientToWindow	IWebBrowser2::put_Height
IWebBrowser2::ExecWB	IWebBrowser2::put_Left
IWebBrowser2::get_Application	IWebBrowser2::put_RegisterAsBrowser
IWebBrowser2::get_Busy	IWebBrowser2::put_RegisterAsDropTarget
IWebBrowser2::get_Document	IWebBrowser2::put_Resizable
IWebBrowser2::get_FullName	IWebBrowser2::put_Silent
IWebBrowser2::get_Height	IWebBrowser2::put_Top
IWebBrowser2::get_HWND	IWebBrowser2::put_Visible
IWebBrowser2::get_Left	IWebBrowser2::put_Width
IWebBrowser2::get_LocationName	IWebBrowser2::PutProperty
IWebBrowser2::get_LocationURL	IWebBrowser2::QueryStatusWB
IWebBrowser2::get_Name	IWebBrowser2::Quit
IWebBrowser2::get_Parent	IWebBrowser2::Refresh
IWebBrowser2::get_Path	IWebBrowser2::Refresh2
IWebBrowser2::get_ReadyState	IWebBrowser2::Stop

The following table shows the enumerations that are related to the Web Browser control that Internet Explorer for Windows CE supports.

BROWSERNAVCONSTANTS	READYSTATE
COMMANDSTATECHANGECONSTANTS	REFRESHCONSTANTS

MLang

MLang is a set of APIs that is designed to help make software that interacts with Internet data more international. More specifically, MLang helps solve problems that are presented by the multilingual environment that exists for software today.

MLang offers APIs for several areas of Internet software internationalization. Included in these APIs is a rich set of character set conversion APIs, the most important of which translates all current Internet character sets to Unicode and back. Developers can use these conversion APIs to base their applications entirely on Unicode and free them from dependencies on particular Internet character sets. MLang also provides two code-page detection methods that automatically determine in which languages and code pages data is written. MLang maps the relationship between a MIME character-set identifier such as Ã ,Â“gb2312Ã,Â” and an unsigned integer that can be passed into MLang APIs or into standard Win32 APIs. MLang provides information about the character set and font support for specific scripts that are available on the computer that is running the software. This script and font information can be used to implement font linking, which is a general way to render an arbitrary Unicode character on a system. To complete the functionality, MLang allows an application to perform globalized line breaking and conversions between language identifiers in RFC1766 format and Microsoft Windows CE locale identifiers (LCIDs).

The following table shows the MLang objects that Windows CE supports.

Code Page Enumeration	MultiLanguage
Conversion	Script Enumeration
Locale	Enumeration

The following table shows the **MLang** interfaces and methods that Windows CE supports.

IEnumCodePage	IMLangFontLink2
IEnumCodePage::Next	IMLangFontLink2::CodePageToScriptID
IEnumCodePage::Reset	IMLangFontLink2::GetFontCodePages
IEnumCodePage::Skip	IMLangFontLink2::GetFontUnicodeRanges
IEnumRfc1766	IMLangFontLink2::GetScriptFontInfo
IEnumRfc1766::Next	IMLangFontLink2::MapFont
IEnumRfc1766::Reset	IMLangFontLink2::ReleaseFont
IEnumRfc1766::Skip	IMLangFontLink2::ResetFontMapping
IMLangCodePages	IMLangLineBreakConsole
IMLangCodePages::CodePagesToCodePage	IMLangLineBreakConsole::BreakLineA
IMLangCodePages::CodePageToCodePages	IMLangLineBreakConsole::BreakLineW
IMLangCodePages::GetCharCodePages	IMultiLanguage
IMLangCodePages::GetStrCodePages	IMultiLanguage::ConvertString
IMLangConvertCharset	IMultiLanguage::ConvertStringFromUnicode
IMLangConvertCharset::DoConversion	IMultiLanguage::ConvertStringToUnicode
IMLangConvertCharset::DoConversionFromUnicode	IMultiLanguage::CreateConvertCharset
IMLangConvertCharset::DoConversionToUnicode	IMultiLanguage::EnumCodePages
IMLangConvertCharset::GetDestinationCodePage	IMultiLanguage::EnumRfc1766
IMLangConvertCharset::GetProperty	IMultiLanguage::GetCharsetInfo
IMLangConvertCharset::GetSourceCodePage	IMultiLanguage::GetCodePageInfo
IMLangConvertCharset::Initialize	IMultiLanguage::GetFamilyCodePage
IMLangFontLink	IMultiLanguage::GetLcidFromRfc1766
IMLangFontLink::GetFontCodePages	IMultiLanguage::GetNumberOfCodePageInfo
IMLangFontLink::MapFont	IMultiLanguage::GetRfc1766FromLcid
IMLangFontLink::ReleaseFont	IMultiLanguage::GetRfc1766Info
IMLangFontLink::ResetFontMapping	IMultiLanguage::IsConvertible

The following table shows the MLang structures that Internet Explorer for Windows CE supports.

MIMECPINFO	MIMECSETINFO	RFC1766INFO

The following table shows the MLang enumerations that Internet Explorer for Windows CE supports.

MIMECONTF	MLCONVCHAR

For More Information

For more information on Web site design and development, see the MSDN Library.

For more information on **WinInet** functions, see the Microsoft Windows CE 3.0 Platform Builder Software Development Kit (SDK) documentation.

For more information on Windows CE development and customizing Windows CE, see the Microsoft Windows CE 3.0 Platform Builder Software Development Kit (SDK) documentation.

For more information on the Microsoft Windows programming environment, see both the Microsoft Windows CE 3.0 Platform Builder SDK and *Programming Windows* by Charles Petzold (Microsoft Press, 1999).

Welcome to the MSDN Library

Using XML and XSL in Pocket Internet Explorer

Get started in using XML, XSL, XML DOM, and JScript in Microsoft Internet Explorer for the Pocket PC as I walk you through some sample code.

Download 745-CF-DEV.exe.

Applies to:
Your current Web application toolset
XML support in Microsoft® SQL Server™ 2000
Microsoft Windows® Powered Pocket PC 2000
Microsoft Pocket Internet Explorer

Gotchas

When you create and edit XML (Extensible Markup Language) and XSL (Extensible Stylesheet Language) files, you have to do it very carefully. Neither XML nor XSL is "forgiving" when it comes to formatting and you might want to use a tool to help you find syntax errors.

Languages Supported

English

Separating Content From Layout

The whole idea with XML is to separate the content from the layout of a Web page. Traditionally, you have used HTML (HyperText Markup Language) to create your Web pages and if you have started to build server-side applications, you have created those HTML pages dynamically using something like ASP (Active Server Pages). As long as you only needed to support a limited number of clients (browsers), you could get away with using a "least denominator" approach (only include things that were supported by all clients/ browsers).

As you may have seen in my article Make Your Web Applications Support Pocket PC, you can add support for multiple clients (I talked about Pocket PC and Microsoft Mobile Explorer in the article). You can do this by dynamically delivering different content depending on which type of client is accessing the ASP page using the HTTP (HyperText Transfer Protocol) information. That is a great way to get started on a multi-channel strategy, but if you want a more solid approach, you should look at XML.

The interesting thing is that nowadays Microsoft has added support for XML in most of its products and what is particularly interesting is that they have added support for XML and even XSL on the Pocket PC. XSL is primarily used to format XML into HTML. The XML/XSL support in Pocket Internet Explorer has many of the important features from the PC implementation.

XML Impact

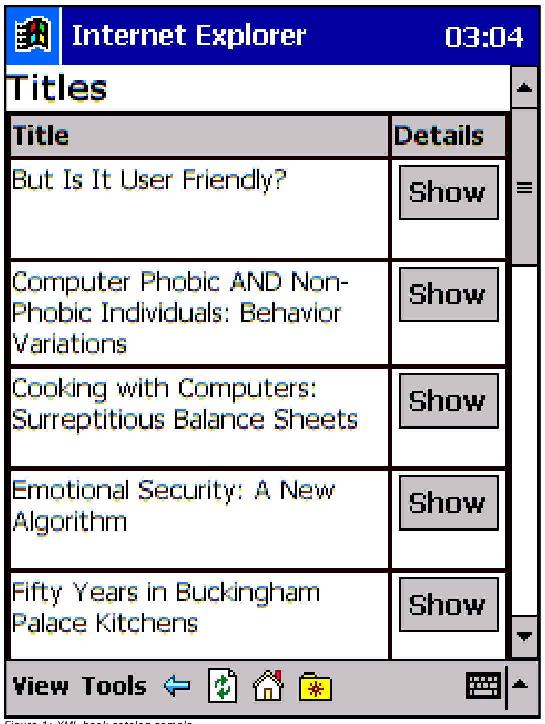
One interesting aspect of separating content (data) from layout is that when the Web page needs to be updated, you only need to transfer the new data to the client. In a wireless scenario, where bandwidth is a serious issue, this is really interesting. Any amount of data that we can avoid passing "in the air" has a positive impact on cost.

Another issue is server load. If we do all the handling of content adapted for different clients, we have to do an awful lot of work on the server. If we would let the server simply provide new data on a "need-for-update" basis, we would not only save valuable server processing power, but also bandwidth at the same time.

Also, when we want to take Web pages offline another issue becomes important—storage space. If you copy all the Web pages of your application to a Pocket PC Web page cache, you will end up with all the layout (formatting) information stored over and over again (in each page).

A Catalog Sample

To show you how the impact for a business application, I have created a sample book catalog based on a table in the pubs database that comes with SQL Server 2000. When the user enters the address to the catalog XML file (titles.xml), this is how it looks:

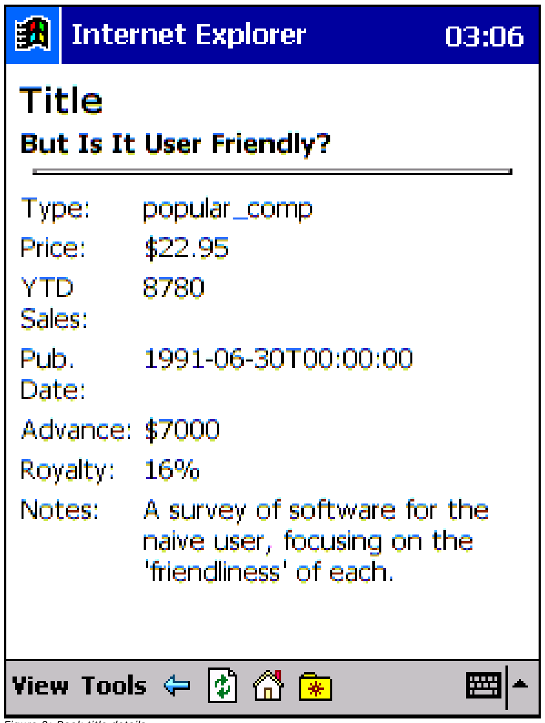


Titles

Title	Details
But Is It User Friendly?	Show
Computer Phobic AND Non-Phobic Individuals: Behavior Variations	Show
Cooking with Computers: Surreptitious Balance Sheets	Show
Emotional Security: A New Algorithm	Show
Fifty Years in Buckingham Palace Kitchens	Show

Figure 1: XML book catalog sample.

And when the user taps on the top **Show** button, this is what she gets:



Title

But Is It User Friendly?

Type:	popular_comp
Price:	$22.95
YTD Sales:	8780
Pub. Date:	1991-06-30T00:00:00
Advance:	$7000
Royalty:	16%
Notes:	A survey of software for the naive user, focusing on the 'friendliness' of each.

Figure 2: Book title details.

This is a fairly simple example of what you can do with XML and XSL and you might say that you could have done this with some server-side scripting as well. And you're right. The interesting thing about this sample is that both views (list and details) use the same source (in XML) for displaying the layout. If you had built the solution in ASP, you would have sent a huge amount of HTML over the wire (or in the air). Another interesting thing is what happens when the book information changes on the server. The field **YTD Sales** is probably something that changes often, and therefore you would probably like to update only the data, as the layout will not change as often.

Another interesting thing about the sample is that even if it's based on XML and XSL, you can take it offline. If you move the files from the server to a folder on your Pocket PC, you can view the same information without the need of any connection. If you would do the same thing for an ordinary Web application, you would probably end up with a lot of static Web pages in your offline Web page cache. In this simple sample, you end up with a demand for almost 20 times (!) as much space on your Pocket PC.

Sample Walk Through

When you start looking at the sample files, you can see that the XML source is a plain file containing all the titles (titles.xml). In a real-world scenario, this file would most probably have been generated from a database source. When I created the file, I used the XML support in SQL Server 2000. After setting up a virtual directory (pubs) for XML access to the **pubs** database, I used the following URL (Universal Resource Locator) to get the XML data:

```
http://localhost/pubs?SQL=select+*+from+titles+for+xml+auto&root=titlelist
```

And if you would like to take a look yourself at out how the XML result is transformed using the XSL file (titlesSQL.xsl), you have to create a normal virtual directory (piexml) in Internet Information Server, place the XSL file (titlesSQL.xsl) there. Then try the following URL:

```
http://localhost/pubs?SQL=select+*+from+titles+for+xml+auto&root=titlelist&xsl=titlesSQL.xsl
```

Both the above URLs work in the PC version of Internet Explorer and Internet Explorer for the Pocket PC. As I have already exported the book data to a XML file (titles.xml), you will not need SQL Server 2000 to try the sample out.

Let's start by looking at part of the titles.xml file (first two book titles):

```
<?xml version="1.0" encoding="utf-8" ?>
<?xml:stylesheet type="text/xsl" href="titles.xsl" ?>
<titlelist>
  <titles title_id="BU1032" title="The Busy Executive&apos;s Database Guide" type="business    " pub_id="1389"
  price="19.99" advance="5000" royalty="10" ytd_sales="4095" notes="An overview of available database systems
with
  emphasis on common business applications. Illustrated." pubdate="1991-06-12T00:00:00"/>
  <titles title_id="BU1111" title="Cooking with Computers: Surreptitious Balance Sheets" type="business    "
  pub_id="1389" price="11.95" advance="5000" royalty="10" ytd_sales="3876" notes="Helpful hints on how to use
your
  electronic resources to the best advantage." pubdate="1991-06-09T00:00:00"/>
.
.
.
</titlelist>
```

You can see that each row in the database is stored as a **titles** XML tag and each column is an attribute on that tag. This may not be the most common way of formatting data in XML, but it is the native format that SQL Server 2000 uses. Even if you would store each column as a sub-tag to the row tag, the sample XSL files wouldn't look much different. On the second line you can see that the XML file points to a XSL file (titles.xsl) to use for formatting.

So, let's go on and look at the titles.xsl file:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="uri:xsl">
  <xsl:template match="/">
    <SCRIPT>
    function showDetail(title_id)
    {
      var root = detailXSL.documentElement;
      var selectedElems =  root.selectNodes("//xsl:if");
      var ifStatement = selectedElems.item(0);
      ifStatement.attributes(0).text =
                      "@title_id[.='" + title_id + "']";

      document.write(titles.transformNode(detailXSL.documentElement));
    }
    </SCRIPT>
    <H3>Titles</H3>
    <TABLE BORDER="1" CELLPADDING="1" CELLSPACING="0">
      <TR>
        <TD BGCOLOR="#C0C0C0"><B>Title</B></TD>
        <TD BGCOLOR="#C0C0C0"><B>Details</B></TD>
      </TR>
      <xsl:for-each select="titlelist/titles" order-by="@title">
        <TR>
          <TD VALIGN="top"><xsl:value-of select="@title"/></TD>
          <TD VALIGN="top"><INPUT>
            <xsl:attribute name="TYPE">button</xsl:attribute>
            <xsl:attribute name="VALUE">Show</xsl:attribute>
            <xsl:attribute name="ONCLICK">showDetail('<xsl:value-of select="@title_id"/>')</xsl:attribute>
          </INPUT></TD>
        </TR>
      </xsl:for-each>
    </TABLE>
    <XML ID="titles" SRC="titles.xml"></XML>
    <XML ID="detailXSL" SRC="detail.xsl"></XML>
  </xsl:template>
</xsl:stylesheet>
```

If you just look at the HTML part for this file, you see that we first define a table with two columns (**Title** and **Details**) and then fill the table with a row for each title in the XML file. We also sort the title list on the title of the book. In the second column of each row, we create a **Show** button that calls the JScript function (showDetail) declared in the top <SCRIPT> tag. This function loads the XSL file for viewing the details page (details.xsl defined in the XML "island" at the bottom), finds the "if" statement, and updates the title ID (title_id) to look for. The function finally transforms the XML file (titles.xml) using the XSL (details.xsl) and writes the output to the HTML document in the browser. The showDetail function uses the XML DOM (Document Object Model) to do its work.

We had better have a look at the details.xsl file too:

```
<?xml version="1.0"?>
```

```
<xsl:stylesheet xmlns:xsl="uri:xsl">
  <xsl:template match="/">
    <H3>Title</H3>
    <TABLE BORDER="0" CELLPADDING="1" CELLSPACING="0">
      <xsl:for-each select="titlelist/titles">
        <xsl:if test="@title_id[.='??????']">
          <TR>
            <TD VALIGN="top" COLSPAN="2"><B><xsl:value-of
              select="@title"/></B><BR/><HR/></TD>
          </TR>
          <TR>
            <TD VAlIGN="top">Type:</TD>
            <TD VALIGN="top"><xsl:value-of select="@type"/></TD>
          </TR>
          <TR>
            <TD VAlIGN="top">Price:</TD>
            <TD VALIGN="top">$<xsl:value-of select="@price"/></TD>
          </TR>
          <TR>
            <TD VAlIGN="top">YTD Sales:</TD>
            <TD VALIGN="top"><xsl:value-of select="@ytd_sales"/></TD>
          </TR>
          <TR>
            <TD VAlIGN="top">Pub. Date:</TD>
            <TD VALIGN="top"><xsl:value-of select="@pubdate"/></TD>
          </TR>
          <TR>
            <TD VAlIGN="top">Advance:</TD>
            <TD VALIGN="top">$<xsl:value-of select="@advance"/></TD>
          </TR>
          <TR>
            <TD VAlIGN="top">Royalty:</TD>
            <TD VALIGN="top"><xsl:value-of select="@royalty"/>%</TD>
          </TR>
          <TR>
            <TD VAlIGN="top">Notes:</TD>
            <TD VALIGN="top"><xsl:value-of select="@notes"/></TD>
          </TR>
        </xsl:if>
      </xsl:for-each>
    </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

At the top of the file, you find the "if" statement (xsl:if) that was manipulated from the showDetail function in the titles.xsl file. And we only show the column values for the selected title in the resulting HTML page.

Conclusion

Hopefully, now you can fully agree with me that there are a number of reasons why you should start looking at XML and XSL for your multi-channel applications. You can save time in designing and implementing your Web applications, and you will also save server load and (wireless) network bandwidth. When taking your XML applications offline, you will also save storage space in your Pocket PC. My advice to you is to start diving into the world of XML and start separating content from layout in your Pocket PC Web applications—right now!

How To Specify a URL When Starting Pocket Internet Explorer from eVB

Article ID : 275230
Last Review : July 13, 2004
Revision : 1.0

This article was previously published under Q275230

On this page

SUMMARY

This article demonstrates how to start Microsoft Pocket Internet Explorer on a Pocket PC with a specific URL from eMbedded Visual Basic (eVB) code.

MORE INFORMATION

To do this, you need to use the **ShellExecuteInfoEx** API function from a dynamic-link library (DLL) created in eMbedded Visual C++ (eVC) and pass in the URL to which you want to browse. Then, you need to write a Declare statement in eVB, and call the function from eVB code.

Create the eVC DLL

1. Start eVC, and from the **File** menu, click **New**.
2. From the **Projects** tab, select **WCE Dynamic-Link Library**.
3. Name the project GetURLDll.
4. In the next screen of the wizard, select **A DLL that exports some symbols** and click **Finish**.
5. Browse to the GetURLDll.cpp file, and add the following code:

```
int WINAPI CallShell(LPCTSTR lpFilePath, LPCTSTR lpURL)
{ SHELLEXECUTEINFO *lpSHInfo; BOOL RetVal; lpSHInfo =
(SHELLEXECUTEINFO *) HeapAlloc(GetProcessHeap(), HEAP_ZERO_MEMORY,
sizeof(SHELLEXECUTEINFO)); lpSHInfo->cbSize = sizeof
(SHELLEXECUTEINFO); lpSHInfo->fMask = SEE_MASK_NOCLOSEPROCESS;
lpSHInfo->lpFile = lpFilePath; lpSHInfo->lpParameters = lpURL; RetVal
= ShellExecuteEx(lpSHInfo); HeapFree(GetProcessHeap(), 0, lpSHInfo);
return RetVal; }
```

6. From the **File** menu, click **New**, and then add a text file named GetURLDll.def to the project. Paste the following code:

```
LIBRARY GetURLDll EXPORTS CallShell @1
```

7. Make sure that the device type is "Pocket PC" and the target is "Win32 (WCE x86em) Release", and then press F7 to build the DLL.

Build the eVB Client

1. Start eVB and select **Windows CE for the Pocket PC Project**.
2. Add a .bas module to the project, and paste the following code:

```
Option Explicit Public Declare Function CallShell Lib "GetURLDll.dll"
_ (ByVal FilePath As String, ByVal DestURL As String) As Integer
Public Sub FindMe() Dim RetVal As Integer Dim Path As String Dim URL
As String Path = "\windows\iexplore.exe" URL = "\windows\calc.htm"
RetVal = CallShell(Path, URL) End Sub
```

3. The compiled DLL needs to be in the \WinCETools\wce300\MS Pocket PC\emulation \palm300 folder, or you will need to use the relative path in the Lib clause of the Declare statement.
4. Add a button to the form, and then paste the following code in the code window of the form:

```
Option Explicit Private Sub Command1_Click() Call FindMe End Sub
Private Sub Form_OKClick() App.End End Sub
```

5. Run the code, and then click the button. The Calculator Help screen appears.

APPLIES TO

- Microsoft eMbedded Visual Basic 3.0

Top of Page

Keywords: kbhowto KB275230

Top of Page

- HOME PAGE
- All Articles
- Advertise
- Consulting

Development
- Discuss - Forums
- Still in the box?
- .Compact Framework
- SQL Server CE
- Code Snippets
- Controls
- MS Resources
- eVB Legacy
- Database
- Personal Media Ctr
- Gizmobility

Stores
- Tablet PC
- Pocket PC Hardware
- Pocket PC Software
- Compact Flash Cards
- Secure Digital Cards
- Smartphone Software
- Pocket PC Books
- .NET CF Books
- Book Reviews
- SPB SW Discounts
- RESCO SW Discounts

DEVBUZZ Info
- About Us
- Help
- Join our email list
- Links & Ratings
- Press & Comments
- Pocket PC version
- Software Reviews
- Hardware Reviews

Authors
- Authors
- Article Guide
- Competitions

Resources
- Developers
- Register
- Login

Development | Starting Out

PocketASP, ASP on your Pocket PC

Written by Vince Singleton [author's bio] [read 25439 times]
Edited by Derek

DISCUSS THE ARTICLE... VERSION: eVB 3.0

Page 1 Page 2

The Background

Give a lazy man the hardest job and he'll find the easiest way of doing it. Now I'm not saying I'm lazy of course, but when I first started looking into developing applications on the Pocket PC platform there's plenty there that makes you think there could be some long nights ahead:

- Maintaining numerous processor specific versions of code
- Worrying about component dependencies
- Getting to grips with the new development environment
- Download and debugging on the device (emulation is never quite the same)

That's just for starters, no doubt there are plenty more items others might like to add to this list. Of course this is all good character (and skill) building stuff, but that might not be how my superiors view it while waiting for a Pocket PC application release date. So, I wanted to work in an efficient and familiar way, but target this new platform...enter ASP. Here is a framework that is used the world over to rapidly develop (web) applications, often hooked up to databases and which supports the VBScript language familiar to millions, not to mention the fact that I've been developing in it for years. If it's good enough for thousands of web applications, then it's good enough for the Pocket PC!

Going through this thought process created the long-term objective for PocketASP. The developer must be able to create, view and debug ASP on their PC as normal (Visual InterDev and IIS for example), only as a final stage should they need to copy it down (unmodified) to the Pocket PC to run final checks.

Spb Software House

Software Development for a Mobile World

Columnists

- Rick Winscot

Past Blast

- Old news
- Hosted Software
- Wireless
- Newsletters
- Carl Davis
- Upton Au

Pocket PC Registry

- Join the registry
- View current list

Current Poll

Are you converting to .NET Compact Framework?

[✓ Yes, it has changed my life!]

No, I'm sticking with eVB

.NET CF what's that`?

Vote!

Current results

2797 votes so far

Recent Forum Threads [goto forums]

OpenNETCF ButtonEx known bug
read... (15 hits)

Verifying server connection vb.net CF??
read... (47 hits)

dynamically changing menu attributes eVC++
read... (55 hits)

Question for Rob
read... (54 hits)

A question for Rob
read... (63 hits)

Access to Excel File under EVB possible ?
read... (86 hits)

Wifi data transfer from PDA to PC
read... (107 hits)

ListBox
read... (135 hits)

Reset Radio Buttons
read... (104 hits)

The Platform

Enough of where PocketASP came from, what can it do in its current release? The evaluation version contains a set of example ASP pages to demonstrate the key supported features, such as:

- Obtain input from the user, using the Form POST and GET processing
- Maintain session information, using the Session object
- Manage inter-session information, using persistent cookies
- Support the usual VBScript language syntax and features such as include files, function calls, sub routines etc.
- Database support, provided via familiar ADO Connection and Recordset objects

The final point, database support, is worth a little expansion as that provides the essential connectivity when creating PC/Pocket PC integrated applications, especially when coupled with the database merging functionality provided by Active Sync 3.1. There are some restrictions, as the underlying technology used on the Pocket PC is ADOCE, but all the essentials are there for creating, reading, updating and deleting database information.

Next Page

Back to Starting Out | **[Article Index]**

TOP

help
read... (120 hits)

Code for Drawing a Line with a Width!
read... (145 hits)

Grouping Radio And/Or CheckBoxes
read... (150 hits)

Protecting Source
read... (163 hits)

A bit of VB.NET help please!
read... (174 hits)

Active Learnings, JADE LEAP / VB.NET
read... (142 hits)

How PPC sync with only one PC?
read... (176 hits)

DON'T LAUNCH!!! (ActiveSync)
read... (180 hits)

Querying Supported Codecs Registry? API?
read... (171 hits)

Quering Audio Drivers
read... (165 hits)

RGB(248, 252, 248)
read... (160 hits)

DEVBUZZ.COM Handheld development!

Creating POOM items using PIE Web pages

SEARCH

▸ HOME PAGE
▸ All Articles
▸ Advertise
▸ Consulting
Development
▸ Discuss - Forums
▸ Still in the box?
▸ .Compact Framework
▸ SQL Server CE
▸ Code Snippets
▸ Controls
▸ MS Resources
▸ eVB Legacy
▸ Database
▸ Personal Media Ctr
▸ Gizmobility
Stores
▸ Tablet PC
▸ Pocket PC Hardware
▸ Pocket PC Software
▸ Compact Flash Cards
▸ Secure Digital Cards
▸ Smartphone Software
▸ Pocket PC Books
▸ .NET CF Books
▸ Book Reviews
▸ SPB SW Discounts
▸ RESCO SW Discounts
DEVBUZZ Info
▸ About Us
▸ Help
▸ Join our email list
▸ Links & Ratings
▸ Press & Comments
▸ Pocket PC version
▸ Software Reviews
▸ Hardware Reviews
Authors
▸ Authors
▸ Article Guide
▸ Competitions
Resources
▸ Developers
▸ Register
▸ Login

Development | Starting Out

Creating POOM items using PIE Web pages
Written by John Cody [author's bio] [read 22177 times]
Edited by Derek

DISCUSS THE ARTICLE... VERSION: eVB 3.0

Page 1 Page 2

Creating POOM items using PIE Web pages

The ability to manipule Pocket Outlook items such as Contacts, Appointments and Tasks from an eVB app is a very cool feature. Once the various items are added into the outlook database, your app could filter and display those items based on the user's preferences, or allow the user to quickly find a specific item. However, the process of creating a new outlook item is usually a manual process. The user typically selects "New" and then tediously enters the various text data of the item using the very small SIP keyboard or by writing on the screen. This is not only a time consuming task, but it is prone to errors.

A "Contact" item is particularly time consuming to enter. You typically end up entering a company name, first and last name, telephone number and an email address before saving it.

Wouldn't it be nice to have all of a contacts' data automatically entered into your Pocket PC simply by clicking on a link of a webpage? Well, that's exactly what I am about to show you how to do.

I could have designed an eVB app to accomplish this, but it would require each user to first install the eVB app on their Pocket PC before they can start adding contacts. This method would also has the disadvantage of requiring the user to install continuous program updates on their Pocket PC to facilitate new features or fix bugs with an existing version.

The method I chose does not require any special software to be installed on the user's Pocket PC, so virtually any Pocket PC can immediately utilize my technique to add new contacts right out of the box. In addition, new features and capabilities can be easily added without a single change to the user's Pocket PC.

Spb Software House

Software Development for a Mobile World

Columnists
- Rick Winscot

Past Blast
- Old news
- Hosted Software
- Wireless
- Newsletters
- Carl Davis
- Upton Au

Pocket PC Registry
- Join the registry
- View current list

Current Poll
Are you converting to .NET Compact Framework?

[✓ Yes, it has changed my life!]
No, I'm sticking with eVB
.NET CF what's that`?

Vote!
Current results
2797 votes so far

Recent Forum Threads [goto forums]

OpenNETCF ButtonEx known bug
read... (15 hits)

Verifying server connection vb.net CF??
read... (47 hits)

dynamically changing menu attributes eVC++
read... (55 hits)

Question for Rob
read... (54 hits)

A question for Rob
read... (63 hits)

Access to Excel File under EVB possible ?
read... (86 hits)

Wifi data transfer from PDA to PC
read... (107 hits)

ListBox
read... (135 hits)

Reset Radio Buttons
read... (104 hits)

The key to my technique is PIE's support of JavaScript. Even though I have never programmed in JavaScript before, it was not too difficult to understand because of a few key similarities it has with Basic.

Let's now dive into the details of my technique...

Because JavaScript is a scripting language, it has many of the same capabilities as VBScript, including the key ability to reference ActiveX Objects. Luckily, the ActiveX control 'PIMSTORE.DLL' is included in every Pocket PC's ROM. This was a critical factor in allowing Pocket PC's to use my technique out-of-the-box (without installing any special software).

NOTES:

1) A download which includes more info and example code for using POOM can be found at: http://www.microsoft.com/mobile/developer/downloads/poomsdk.asp

2) The code below was reported to work on first-generation Casio, HP and iPaq Pocket PC's. However, it appears not to work on the newer Pocket PC's 2002. I am waiting for the PC 2002 upgrade for my test 3135 mono iPaq so I can find out what's happening. It is probably something simply like a slight syntax change such as removing the "()" after calls such as "pol.logon" - which desktop IE requires to work.

The first step in creating a new pocket outlook Contact item, is to create an instance of the Pocket Outlook Object Module (POOM). This is accomplished from within JavaScript as shown below:

```
var pol;
pol = new ActiveXObject("pocketoutlook.
application");
```

```
Next, we need to log into the Pocket Outlook
Datastore:
```

```
pol.logon();
```

Then, we simply create a reference to a new Contact item:

```
contact = pol.createitem(2);
```

```
Set the various fields of the new contact item:
```

```
contact.CompanyName = "deVBuzz.com";
contact.Email1Address = "derek@devbuzz.com";
contact.FirstName = "Derek";
contact.LastName = "Mitchell";
contact.webpage = "http://www.devbuzz.com/pie";
```

help
read... (120 hits)

Code for Drawing a Line with a Width!
read... (145 hits)

Grouping Radio And/Or CheckBoxes
read... (150 hits)

Protecting Source
read... (163 hits)

A bit of VB.NET help please!
read... (174 hits)

Active Learnings, JADE LEAP / VB.NET
read... (142 hits)

How PPC sync with only one PC?
read... (176 hits)

DON'T LAUNCH!!! (ActiveSync)
read... (180 hits)

Querying Supported Codecs Registry? API?
read... (171 hits)

Quering Audio Drivers
read... (165 hits)

RGB(248, 252, 248)
read... (160 hits)

```
contact.body = "deVBuzz.com is dedicated to …";
```

Save it and log off:

```
contact.save();
pol.logoff();
```

The lines of code above represent the raw code of my technique. But to be truly useful, it needs to reside in a web page/HTML file. Below is a complete HTML file incorporating my technique and even a little extra code to check to see if the contact already exists in the user's database, so it won't add a duplicate entry.

```
<head><title>Poom Test</title></head>
<body topmargin="0">
&nbsp;<p align="center"><b>Creating Outlook Items
via<br>Pocket Internet Explorer<br>
Webpages</b></p>
<p align="center">&nbsp;</p>
<p align="center">By John Cody<br>and deVBuzz.
com&nbsp;</p><hr>
```

```
<p align="center"><font size="3"><a href="#"
onclick="addcontact()"><br>Click Here<br>
```

```
</a>To add <b>deVBuzz.com<br>
</b>to your Contact list.
</font></p></body>
```

Next Page

Back to Starting Out | [**Article Index**]

TOP

CONTACT US PRESS
ABOUT
English

Spb Software House

SOFTWARE DEVELOPMENT FOR A MOBILE WORLD

POCKET PC
SMARTPHONE
DOWNLOADS
ENTERPRISE
SUPPORT

Spb Newsletter

your@email.com

Spb Software House Releases Spb Clone
(January 25, 2005)
Spb extends the enterprise product line by releasing Spb Clone - an application for multiple Pocket PC device duplication. Configure one device and clone it hundreds times!
More...

Spb Software House Updates Spb Finance and Adds Localized Versions
(January 17, 2005)
All the three editions of Spb Finance get now localized for 6 languages (English, German, French, Italian, Russian, and Spanish). Manage your finances in your native language!
More...

Spb Software House Adds Money Synchronization to Spb Finance
(December 21, 2004)
Inspired by the success of the previously released editions of Spb Finance, Spb is now targeting the needs of those whose favorite desktop financial program is Microsoft Money by offering Spb Finance Money Edition.
More...

Spb Finance
NEW! Your Pocket Finance Manager
• Budgets
• Reports
• Multicurrency
More...

Spb Full Screen Keyboard
The ideal solution for touch-screen typing
• Full screen layout
• Large keys
• Word completion
More...

Spb GPRS Monitor
Control your traffic, save your money
• Taskbar icon
• Pop-up window
• Text reports
More...

Spb Imageer
View, Edit & Share Pictures, Anywhere!
• Picture editing
• Batch optimizing
• Web publishing
More...

Spb Pocket Plus
Making the Best Even Better
• Today Plug-in
• Real Close Button
• Battery Indicator
More...

Spb Time
It's Your Time!
• Screensaver
• Analog/Digital clock
• Skins support
More...

Spb Pocket PC Tips & Tricks
200+ tricks and secrets from Pocket PC gurus
• 200+ tips and tricks
• 10 professional authors
• Reveal all the Pocket PC secrets
More...

Spb Clone
Multiplicate your enterprise devices
• Clone image on SD card
• Fast solution deployment
• Solution support
More...

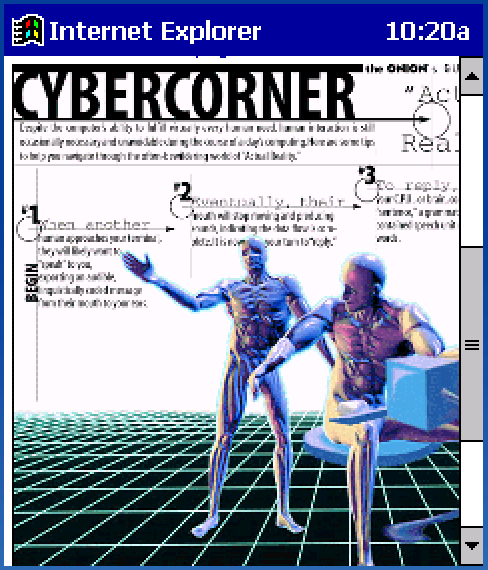
Pocket PC | Smartphone | Downloads | Enterprise | Support About | Contact Us | Press

Microsoft Tech·Ed 2000

it's time to build the business internet

today

Microsoft Internet Explorer and Web-based Applications *for Pocket PC*



Mobile Devices Division
Microsoft Corporation

Today’s Presentation

- Pocket IE Overview
- Accessing data with Pocket IE
  - Online and offline data
  - Delivery mechanisms
- Pocket IE Capabilities
  - Dos and don’ts

What is Pocket IE?

- A best-of-breed browser for the category
  - Full-featured; 3rd-generation browser
  - Optimizations for screen size
- Two ways to use Pocket IE
  - Connected – live TCP/IP
  - Disconnected – cached data

Pocket IE
Connected: Get on the Web

How can I connect?

- Compact Flash modem
- Compact Flash Ethernet
- Digital Phone Card

  (CF Adapter to cell phone)
- IR to cell phone

Microsoft Tech·Ed 2000

Pocket IE
Connected: Get on the Web

What can I do?

- Full browsing – type an URL and go
- Proxy support!
- Content is cached for offline access
  - Favorites indicates whether available
- Secure access with SSL
- Fit-to-screen mode, or virtual 640 x 480 display

Pocket IE
Disconnected: Grab N' Go

AvantGo

- Partnership with Microsoft
- Access directly through Pocket IE
- Author-defined access to data
- Enterprise server solutions
- AvantGo.com

Pocket IE
Disconnected: Grab N’ Go

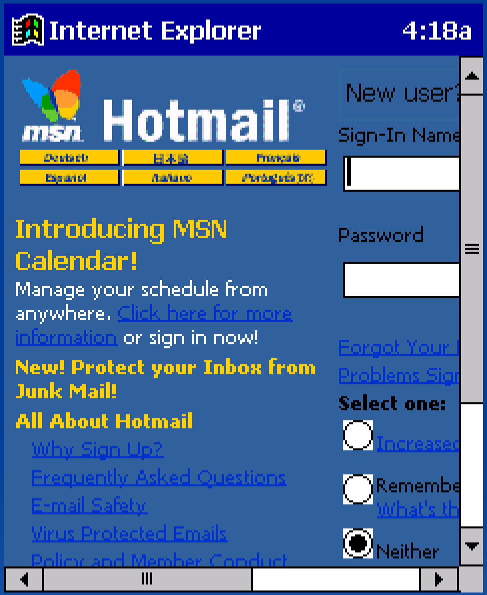
Mobile Favorites

- User-driven access
- Desktop Sync with IE5
- IE5 Plug-in for easy access
- Superset of Mobile Channels

Pocket IE Capabilities

Overview

- HTML 3.2 Compliant
- JavaScript 1.1 compliant
- XML Object Model
- SSL
- Active X support



Introducing MSN Calendar!

Manage your schedule from anywhere. Click here for more information or sign in now!

New! Protect your Inbox from Junk Mail!

All About Hotmail

Why Sign Up?

Frequently Asked Questions

E-mail Safety

Virus Protected Emails

Policy and Member Conduct

New user

Sign-In Name

Password

Forgot Your

Problems Sign

Select one:

Increased

Remembe
What's th

[✓ Neither]

HTML Capabilities
What we do:

- HTML 3.2 support
  - Lightweight & ubiquitous
- Framesets
  - Per HTML 4.0 spec
  - Borders always visible
- Background Images & Sounds

HTML Capabilities
What we don't:

- DHTML
  - Useful on the desktop, but still heavyweight for handhelds
- CSS
  - Can use XSL stylesheets instead
- Animated GIFs

JScript Capabilities

- HTML 3.2-based object model
  - Not the IE4 OM
- Core script support:
  - Scripting against FORM elements
  - Scripting against the XML OM

JScript Capabilities

Not supported:

- Dynamic frameset creation
- Dynamic script generation
- Window.open

Detecting Pocket IE

Server-side VBScript

```
'Check for Windows CE
if (InStr(Request.ServerVariables("HTTP_USER_AGENT"),
    "Windows CE")) then
    { add Windows CE specific code }
else
    { add code for other platforms }
end if
'Check for Pocket PC
if (InStr(Request.ServerVariables("HTTP_UA_OS"),
    "POCKET PC")) then
    { add Pocket PC specific code }
else
    { add code for other platforms }
end if
```

Detecting Pocket IE

Client-side JScript

```
var strNav = navigator.userAgent;
var isCE = strNav.indexOf("Windows CE");
if(isCE > -1) {
    { add Windows CE specific code }
}
else {
    { add code for other platforms }
}
var isPPC = strNav.indexOf("240x320");
  if(isPPC > -1) {
    { add Pocket PC specific code }
}
else {
    { add code for other platforms }
}
```

Microsoft Tech·Ed 2000

XML Capabilities

- Same XML component as IE5
- Markup and transfer of data as XML
- How it works:
  - Data-as-XML delivered from server embedded in HTML page - an XML ‘Data Island’
  - Data read out of page, parsed, and placed into a data tree
  - JScript accesses the XML OM and manipulates the data

XML Capabilities

- Render the XML data in the browser
  - Use XSL to transform XML into HTML
- Describe appearance of HTML page with XML
  - Instead of CSS, markup page with XSL
  - Minimize round-trips to the server
  - Easy support for multiple browser types

Using the XMLHTTP Object

the request

```
<SCRIPT LANGUAGE="JSCRIPT">
var xmlhttp = new ActiveXObject
    ("Microsoft.XMLHTTP");
xmlhttp.Open("POST", "XMLlog.asp", false);

var strXML = "<changeprice SKU='" + ➲
   document.forms[0].SKU.value + "' ➲
   Price='" + iNewPrice + "'/>";

// Send request to logging page
          xmlhttp.Send(strXML);

// Show response (success or failure)
          alert(xmlhttp.responsetext);
</SCRIPT>
```

Microsoft Tech·Ed 2000

Using the XMLHTTP Object

the response

```
<%@ LANGUAGE="VBSCRIPT" %>
<% Response.ContentType = "text/xml"
set XMLReq = Server.CreateObject("Microsoft.XMLDOM")
XMLReq.load(Request)

set xmlAction = XMLReq.selectSingleNode("//changeprice")
iSKU = xmlAction.GetAttribute("SKU")
iPrice = xmlAction.GetAttribute("Price")
{ open recordset containing the requested SKU }
if not rsData.eof then
    rsData("Price") = iPrice
    rsData.update
    response.write "Price changed successfully to $" &
    iPrice & "."
else
    response.write "No record found for this SKU."
end if %>
```

Security Capabilities

For secure transactions, Pocket IE supports:

- NTLM
- SSL
    - 64-bit certificates
    - 128-bit encryption (add-on)
- Others (SGC)

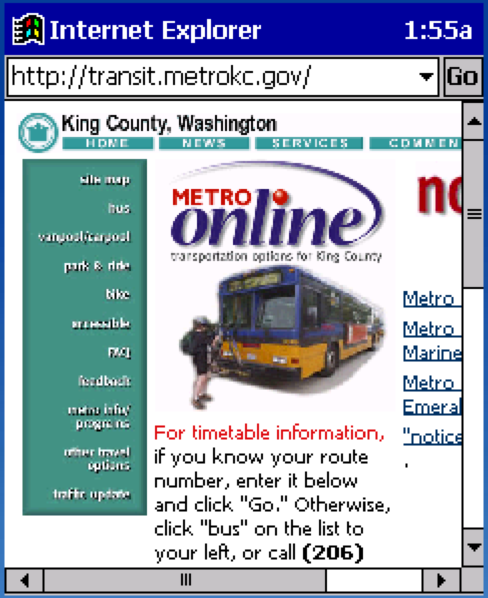
ActiveX Capabilities

- Straightforward – just like the desktop
  - COM component accessed from <OBJECT> tag
  - Script can call ActiveX components, but not vice-versa
- Must be installed on device directly
  - No auto-download
  - Great hook for rich device-web interaction scenarios

Disconnected Data

IE5 Synchronization

- Enables access to any web page when disconnected
- Superset of IE4 channels functionality



For timetable information, if you know your route number, enter it below and click "Go." Otherwise, click "bus" on the list to your left, or call (206)

Disconnected Data
Mobile Favorites

- IE5 Introduces ‘Offline Favorites’
- New ‘Favorites’ sync provider
  - Syncs a subset of your desktop’s web cache with your device’s web cache
  - IE Plugin makes this as easy
    - Grab ‘n Go web pages …

Disconnected Data

What is AvantGo?

- Expandable Solutions
  - Workforce Automation
  - Mobile Navigation
  - Integrated Feedback
- AvantGo.com
  - News (BBC to Financial Times)
  - Service (FedEx, Weather, LastMinute.com)
  - Games (Sony)

Disconnected Data
AvantGo on Pocket PC

Pocket PC is the premiere platform for AvantGo-based solutions

- Color Displays

  Provide higher contrast, greater fidelity

- Large screens

  More data displayed, faster scanning, less scrolling

- Integrated with the browser

  Seamless connected-disconnected use

- Zero install
  - *Ready to run out-of-the-box*

Efficient Use of Pocket IE

Suggestions

- Keep it simple – just the essentials
  - Avoid >2 frames per page
  - Use tables sparingly; allow for dynamic resizing
- Single-column format
  - Pocket IE pages should *never* require horizontal scrolling
  - Makes single-handed reading easy

Guidelines
For more information

- Pocket IE style guide
  http://www.pocketpc.com
- AvantGo style guide
  http://avantgo.com/builder/

Where do you want to go today?®

Internet Explorer for Pocket PC – HTML and Object Model Reference

Last updated 5/09/00

Assumptions and Conventions

All descriptions in this document apply to the Internet Explorer for Pocket PC browser, referred to throughout this document as ‘Pocket IE’.

This document serves as a comprehensive reference for the both rendering and scripting support in Pocket IE. Everything listed in this document is an *element*, an *object*, or both.

> *Elements*, also referred to as *tags*, are described in HTML and determine the appearance of an HTML page.
>
> *Objects* assist page authors who wish to program HTML pages by adding computer code in the form of script. For instance, the *document* object allows script to determine, and change, which page the browser is looking at.
>
> Both: certain elements in HTML are automatically avaiailable to script as objects, and are denoted as an *element/object.*

All *elements, objects,* and *element/objects* are listed in alphabetical order for easy reference.

For all the HTML tag attributes, if the attribute has a value, it is listed as a variable name with a psuedo-Hungarian notation. For instance:

> *sURL* - indicates a variable of type *string,* which in this case contains an URL
>
> *iValue* - indicates a variable of type *integer*
>
> other Hungarian notation:
>
> *bFoo* - variable Foo is of type *boolean*
> *oFoo* - variable Foo is of type *object*
> *vFoo* - variable Foo is of type *variant*

Some tags take no values, for instance the CHECKED attribute for <input type=text>. The omission of any variable name means that it doesn’t take a value.

In the object model portions of the documentation, placeholders for the object are italicized. For instance, *select.*focus() means that a reference to an appropriate object select object should go in place of *select.*

Supported Elements and Objects

<A> element/object

Designates the start or destination of a hypertext link. The A element is an inline element and requires a closing tag.

Attributes

HREF - sURL: string that specifies that destination URL or anchor point
ID - sName: same as NAME; in the event of duplicate NAMEs and IDs, Pocket IE will navigate to the first instance of a particular NAME or ID
NAME - sName: string that specifies a name/bookmark for the current section of the document
TARGET - sTarget: specifies a target frame for the link.
Pocket IE supports special values _top and _parent. The special values: _self and _blank are not supported. _top causes Pocket IE to navigate the "topmost" document to the HREF. _parent causes the immediate parent of the frame to navigate to the HREF.
If target is specified and that frame name doesn't exist (or no target is specified), Pocket IE will navigate the frame that the link was tapped on to the HREF.

Properties

hash - sHash: read/write property that sets or retrieves the part of the HREF after the # mark
host - sHost: read/write property that sets or retrieves the host name part of the URL
hostname – sHostname: read/write property that sets or retrieves the host and domain name or the numeric IP address
href – sHref: read/write property that sets or retrieves the destination URL or anchor point
pathname – sPathname: read/write property that sets or retrieves the filename or path specified by the link
port – sPort: read/write property that sets or retrieves the port specified by the link. The default values are 21 for the ftp protocol, 70 for the gopher protocol, 80 for the http protocol, and 443 for the https protocol.
protocol – sProtocol: read/write property that sets or retrieves the protocol portion of the URL. It returns the initial substring of a URL including the colon.
search – sSearch: read/write property that sets or retrieves the search (query) string portion of the href. This includes the leading question mark.
target – sSearch: read/write property that sets or retrieves the target of the href.

Methods

none

Events

onclick - fires when the user taps on the <A> element. Note the event only fires if the user does a pen down while on the <A> and a pen up without moving >5 pixels away from pen down.   The author can specify special handler for this event.

Collections

none

<ADDRESS> element

Renders its contents in italics. Used to specify information such as the address, signature and authorship for the document. ADDRESS is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

anchors collection

This collection is not implemented in Pocket IE. The length is always 0.

Attributes

none

Properties

length: always returns 0

Methods, Events, Collections

none

<AREA> element/object

Defines the shape, coordinates, and associated URL of one hyperlink region within a client-side image MAP. The AREA element is not rendered and requires a closing tag.

Attributes

COORDS – sCoords: specifies the coordinates of the hyperlink area within an image MAP The format of the string depends on the SHAPE specified.

> SHAPE= "circ" or "circle" COORDS= "x1,y1,r" – Where x1,y2 are the coordinates of the center of the circle, and r is the radius of the circle.
>
> SHAPE= "poly" or "polygon" COORDS= "x1,y1,x2,y2...xn,yn" – Where each x,y pair contains the coordinates of one vertex of the polygon.
>
> SHAPE= "rect" (default) or "rectangle" COORDS= "x1,y1,x2,y2" – Where x1,y1 are the coordinates of the upper-left corner of the rectangle and x2,y2 are the coordinates of the lower-right coordinates of the rectangle.

Note: *percentage values for SHAPE COORDS are not supported.*

If two or more regions overlap, the region defined first in the map definition takes precedence over subsequent regions. This means that AREA elements with NOHREF should generally be placed before ones with the HREF attribute.

HREF- sURL: string that specifies that destination URL or anchor point
NOHREF – specifies that the region has no action, used to exclude areas in an image map
SHAPE – sShape: specifies the shape of an image map region, possible values are circ, circle, poly, polygon, rect, and rectangle.
TARGET - sTarget: specifies a target frame or window for the link.

Pocket IE supports special values _top and _parent.   The special values: _self and _blank are not supported. _top causes Pocket IE to navigate the “topmost” document to the HREF. _parent causes the immediate parent of the frame to navigate to the HREF.

If target is specified and that frame name doesn’t exist (or no target is specified), Pocket IE will navigate the frame that the link was tapped on to the HREF.

Properties

hash - sHash: read/write property that sets or retrieves the part of the HREF after the # mark
host - sHost: read/write property that sets or retrieves the host name part of the URL
hostname – sHostname: read/write property that sets or retrieves the host and domain name or the numeric IP address
href – sHref: read/write property that sets or retrieves the destination URL or anchor point
pathname – sPathname: read/write property that sets or retrieves the filename or path specified by the link
port – sPort: read/write property that sets or retrieves the port specified by the link. The default values are 21 for the ftp protocol, 70 for the gopher protocol, 80 for the http protocol, and 443 for the https protocol.
protocol – sProtocol: read/write property that sets or retrieves the protocol portion of the URL. It returns the initial substring of a URL including the colon.
search – sSearch: read/write property that sets or retrieves the search (query) string portion of the href. This includes the leading question mark.
target – sSearch: read/write property that sets or retrieves the target of the href.

Methods, Events, Collections

none

<B> element

Specifies that the text should be rendered in bold. The B element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<BASE> element

Specifies an explicit URL used to resolve links and references to external source such as links and images. The BASE element does not require a closing tag.

Attributes

HREF – sHref: sets the relative baseline URL on which relative links/references are based. For example:

```
<html>
  <head>
    <base href="http://pocketpc.com/browserweb/">
  </head>
  <body>
    <img src="images/screen.gif">
      <ahref="FULL/graphs/latest.htm">
       Click here to see the latest graph.</a>
  </body>
</html>
```

The SRC will resolve to will resolve to http://pocketpc.com/browserweb/images/screen.gif and the HREF will resolve to will resolve to http://pocketpc.com/browserweb/FULL/graphs/latest.htm.

TARGET – sTarget: specifies a baseline target frame for the page
Pocket IE supports special values _top and _parent. The special values: _self and _blank are not supported. _top causes Pocket IE to navigate the "topmost" document to the HREF. _parent causes the immediate parent of the frame to navigate to the HREF.
If target is specified and that frame name doesn't exist (or no target is specified), Pocket IE will navigate the frame that the link was tapped on to the HREF

Properties, Methods, Events, Collections

none

<BASEFONT> element

Sets attributes of default font to be used when rendering text. This does not require a closing tag.

Attributes

FACE - sFace: specifies a comma separate list of font names that the text should be rendered in
SIZE - iSize: Integer that between 1 and 7 that specifies that font size. The default value is 3.

Properties, Methods, Events, Collections

none

<BGSOUND> element

Enables a background sound to be played when the page is visited. BGSOUND does not require a closing tag.

Attributes

LOOP: not supported in Pocket IE
SRC - sSrc: Specifies that URL of the sound to be played. Not all devices support all sampling rates and bit depths, so this may result in the file not playing. (The only supported file format is .WAV. .MID, .AU. and .AIFF files are not supported.)

Properties, Methods, Events, Collections

none

<BIG> element

Specifies that the enclosed text should be in a larger font than the current font. BIG is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<BLOCKQUOTE> element

Sets apart a quotation in text. Text within the tag is indented. BLOCKQUOTE is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<BODY> element/object

Denotes the beginning and end of the document body. BODY is a block element and requires a closing tag.

Attributes

BACKGROUND - sURL: specifies the URL of a background image to be tiled behind the images and text on a page.
Note: *To improve readability, background images are disabled on grayscale devices and will not render.*
BGCOLOR - sColor: specifies the background color of the page. Values are specified in the format #RRGGBB where RR, GG and BB are hexadecimal values for red, green, and blue levels or can be specified by using one of the following color names listed below.
Note: *Background colors on the body are disabled on grayscale devices. The page will always have a white background.*

Color names:

aliceblue
antiquewhite
aqua
aquamarine
azure
beige
bisque
black
blanchedalmond
blue
blueviolet
brown
burlywood
cadetblue
chartreuse
chocolate
coral
cornflowerblue
cornsilk
crimson
cyan
darkblue
darkcyan
darkgoldenrod
darkgray
darkgreen
darkkhaki
darkmagenta
darkolivegreen
darkorange
darkorchidg
darkred
darksalmon
darkseagreen
darkslateblue
darkslategray
darkturquoise
darkviolet
deeppink
deepskyblue
dimgray
dodgerblue
firebrick
floralwhite
forestgreen
fuchsia
gainsboro
ghostwhite
gold
goldenrod
gray
green
greenyellow
honeydew
hotpink
indianred
indigo
ivory
khaki
lavender
lavenderblush
lawngreen
lemonchiffon
lightblue
lightcoral
lightcyan
lightgoldenrod-
yellow
lightgreen
lightgrey
lightpink
lightsalmon
lightseagreen
lightskyblue
lightslategray
lightsteelblue
lightyellow
lime
limegreen
linen
magenta
maroon
medium-
aquamarine
mediumblue
mediumorchid
mediumpurple
mediumseagreen
mediumslateblue
medium-
springgreen
mediumturquoise
mediumvioletred
midnightblue
mintcream
mistyrose
moccasin
navajowhite
navy
oldlace
olive
olivedrab
orange
orangered
orchid
palegoldenrod
palegreen
paleturquoise
palevioletred
papayawhip
peachpuff
peru
pink
plum
powderblue
purple
red
rosybrown
royalblue
saddlebrown
salmon
sandybrown
seagreen
seashell
sienna
silver
skyblue
slateblue
slategray
snow
springgreen
steelblue

tan	turquoise	whitesmoke
teal	violet	yellow
thistle	wheat	yellowgreen
tomato	white	

LEFTMARGIN - sMargin: specifies the left margin of the body in pixels. Negative values are allowed. The default value for Pocket IE is 6.
LINK - sColor: specifies the color for links on a page. Pocket IE does not distinguish between visited and unvisited links. See BODY BGCOLOR for the format and possible color names.
Note: *Link colors are disabled on grayscale devices resulting in link colors always being black.*
RIGHTMARGIN - sMargin: specifies the right margin of the body in pixels. Negative values don't seem to work. The default value for Pocket IE is 6.
TEXT - sColor: specifies the color of the text on a page. See BODY BGCOLOR for the format and possible color names.
Note: *Text colors are disabled on grayscale devices resulting in text always being black.*
TOPMARGIN - sMargin: specifies the top margin of the body in pixels. Negative values don't seem to work. The default value for Pocket IE is 6.

Properties, Methods

none

Events

onload - event which fires right after the page is loaded
onunload - event which fires right before the current page is unloaded (for example, if the user navigates away from the page or refreshes the page)

Collections

none

<BR> element

Inserts a line break. This element does not require a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<CAPTION> element

Specifies a brief description/caption for a table. The CAPTION element is a block element and requires a closing tag.

Attributes

ALIGN - sAlign: specifies the alignment of the caption with respect to the table. Possible values are: top (default) and bottom.
Note: *VALIGN=top | bottom is not supported.*

Properties, Methods, Events, Collections

none

<CENTER> element

Centers subsequent text and images. This element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<CITE> element

Specifies a citation. The enclosed text is rendered in italics. The CITE element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<CODE> element

Specifies a code sample. The enclosed text is rendered in a mono-spaced font. The CODE element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<COL> & <COLGROUP> elements

These tags are **not** supported in Pocket IE.

<DD> element

Indicates a definition in a definition list (DL). The definition is indented from the definition list. This element is a block element and does not require a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<DFN> element

Indicates the defining instance of a term. The enclosed text is rendered in italics. This element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<DIR> element

Denotes a directory list. The DIR element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<DIV> element

Specifies a container/division in the document. This is a block element and requires a closing tag.

Attributes:

ALIGN - sAlign: specifies the alignment of the DIV. Possible values are left, center, and right.

Attributes, Properties, Methods, Events, Collections

none

<DL> element

Denotes a definition list. The DL element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

document object

Represents the HTML document in a given browser window/frame.

Attributes

none

Properties

alinkColor - sColor: read only property that contains the active link color in the specified document.
Note: *Since Pocket IE doesn't support alinkcolors, this always returns 000000.*
bgColor – sColor: read/write property that sets or retrieves the background color of the specified document. See BODY BGCOLOR for the format and possible color names.
Note: *the color values returned by Pocket IE are <u>not</u> prefixed with a '#'.*
cookie - sCookie: read/write property that sets or retrieves the string value of a cookie. sCookie is the string that specifies or receives the name=value pairs, plus any of the following values:

> *expires=date;*
>
> > Setting no expiration date on a cookie causes it to expire when the browser closes. If you set an expiration date in the future, the cookie is saved across browser sessions. If you set an expiration date in the past, the cookie is deleted. Use GMT format to specify the date.
>
> *domain=domainname;*
>
> > Setting the domain of the cookie allows pages on a domain made up of more than one server to share cookie information.
>
> *path=path;*

Setting a path for the cookie allows the current document to share cookie information with other pages within the same domain—that is, if the path is set to /thispathname, all pages in /thispathname and all pages in subfolders of /thispathname can access the same cookie information.

secure;

Setting a cookie as secure; means the stored cookie information can be accessed only from a secure environment.

domain - sDomain: read only property that contains the domain the document is on, which is initially set to the host that the document is on (like www.pocketpc.com).
fgColor – sColor: read/write property that sets or retrieves the foreground (text) color of the document.
lastModified – -sModified: read only property that contains the time and date the document was last modified
linkColor – sColor: read/write property that sets or retrieves the color of the links in the document.
location – read/write property that sets/retrieves the current URL of the document. Also, is an object that represents information about the current document. See location object for more details.
referrer - sReferrer: read only property that is supposed to contain the URL of the document where the current document was linked from.
Note: *In Pocket IE, this property will always it always return an empty string.*
title – sTitle: read only property that contains the TITLE of the document.
URL - sURL: read only property that contains the URL of the current document.
vlinkColor – sColor: read only property that contains the color of visited links in the specified document.
Note: *Since Pocket IE doesn't support vlinkcolors, this property always returns 000000.*

Methods

clear - syntax: *document*.clear(): Does the same thing as document.open(). See open method below for more details.
close - syntax: document.close(): Closes output stream and forces the data written to the document to be displayed.
open - syntax: *document*.open(): not supported in Pocket IE.
write - syntax: *document*.write(sText): Writes sText (which can contain HTML) to the specified document.
writeln - syntax: *document*.writeln(sText): Writes sText (which can contain HTML) followed by a carriage return to the specified document. ***Note:*** *carriage returns are ignored in HTML unless they're within preformatted text.*

Events

none

Collections:

anchors - collection which is supposed to contain an array of anchors in objects the document. However, the anchor object is unimplemented in Pocket IE, so elements of this collection are all null.

forms - collection of all the form elements on the page. See forms collection for more information.
links – collection of all the <A> elements with the HREF attribute and <AREA>s. See links collection for more information.

<DT> element

Denotes a definition term within a definition list (DL). The DT element is a block element and does not require a closing tag.

Attributes, Properties, Methods, Events, Collections

none

elements collection

Collection of all the controls in a given form. Note that for Pocket IE input type=image controls are included in this collection.
syntax:
[*colElement*s=]*form*.elements
[*oObject*=]form.elements(vIndex)

colElements:
Collection of controls
oObject:
Reference to an individual item in the array of elements contained by the object.
vIndex:
Required. Integer that specifies the element or collection to retrieve.
***Note:** the value of vIndex cannot be accessed as a string.*

Properties

length - iLength: read only property that retrieves the number of elements in the elements collection.

Methods, Events, Collections

none

<EM> element

Emphasizes text by rendering it in italics. The EM element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<FONT> element

Specifies a new font, size, and color to be used for rendering the enclosed text. The FONT element requires a closing tag.

Attributes

COLOR - sColor: sets the color that the text should be rendered in. See BODY BGCOLOR for the format and possible color names. Note: Font colors are intentionally disabled on grayscale devices resulting in text always being rendered in black.
FACE - sFace: specifies a comma separate list of font names that the text should be rendered in
SIZE - iSize: Integer that between 1 and 7 that specifies that font size. The default value is 3. It also can be a relative font size with a signed integer value, e.g. size=”+1”, or size=”-2”. This is mapped to an absolute font size by adding current BASEFONT SIZE.

Properties, Methods, Events, Collections

none

<FORM> element/object

Specifies that the contained controls are part of a form. The FORM element is a block element and requires a closing tag.

Attributes

ACTION - sURL: specifies the URL to which the FORM content is to be sent for processing
METHOD - sMethod: specifies how the form data is sent to the server. Possible values:
get:

append the arguments to the action URL and open it as if it were anchor
***Note:** If the ACTION contains a question mark and a query string, when submitted Pocket IE sends the action URL (including its question mark and query string) along with the arguments.*

post:

send the data thru an HTTP post transaction

NAME - sName: specifies the name for the FORM so that it can be referred to from script.
TARGET - sTarget: specifies a target frame or window for the results of the form submission.
Pocket IE supports special values _top and _parent. The special values: _self and _blank are not supported. _top causes Pocket IE to navigate the “topmost” document to the HREF. _parent causes the immediate parent of the frame to navigate to the HREF.
If target is specified and that frame name doesn’t exist (or no target is specified), Pocket IE will navigate the frame that the link was tapped on to the HREF.

Properties

action – sAction: read/write property that sets or retrieves the URL to which the FORM content is sent for processing
encoding – sEncoding: read only property which retrieves the MIME encoding for the form. The value is always “application/x-www-form-urlencoded”.
method – sMethod: read/write property which specifies how the form data is sent to the server. The only valid values are “post” and get”.

name – -sName: read only property that retrieves the name of the FORM.
target – -sTarget: read/write property that sets or retrieves the TARGET of the FORM. See FORM TARGET attribute for the possible values and behavior.

Methods

reset - syntax: *form*.reset(): simulates a mouse click on the reset button which resets the form back to its initial state.
submit - syntax: *form*.submit(): submits the form but does NOT fire the onsubmit event.

Events

onsubmit - fires when a form is about to be submitted. ***Note:*** *this event only fires when the submit is initiated by the user. It will not fire if the submit() method is called on the form.*

Collections

elements - Collection of all the elements in the form. See elements collection for more information.

forms collection

Collection of all FORM objects in the document in source order.
syntax:
  [*colForms*=]*document*.forms
  [*oObject*=]*document*.forms(*vIndex*)

colElements:
  Collection of forms
oObject:
  Reference to an individual item in the array of elements contained by the object.
vIndex:
  Required. Integer that specifies the element or collection to retrieve.
  Note: *the value of vIndex cannot be accesed as a string.*

Attributes

none

Properties

length - iLength: read only property that contains number of form objects in the collection.
Note: *To find the number of elements in a particular form object, use forms[index].elements.length, not forms[index].length.*

Methods, Events, Collections

none

<FRAME> element/object

Specifies an individual frame within a FRAMESET. FRAME is a block element and does not require a closing tag.

Attributes

MARGINHEIGHT - iHeight: specifies the top and bottom margins for the contents of the frame. The default value for MARGINHEIGHT is 6.
MARGINWIDTH - iWidth: specifies the left and right margins for the contents of the frame. The default value for MARGINWIDTH is 6.
NAME - sName: specifies the name of the frame
NORESIZE - bResize: specifies whether or not the user can resize the frame.
Note: *to allow users to manage frames when browsing on the Pocket PC screen, this attribute is ignored by Pocket IE.*
SCROLLING - sScrolling: specifies whether or not the user can scroll the frame.
Supported values:
auto (default):
    browser determines if scrollbars are necessary
yes:
    frame can be scrolled
SRC - sSRC: specifies the URL to be loaded in the FRAME

Properties, Methods, Events, Collections

none

frames collection

Retrieves a collection of all window objects defined by a given document or defined by the document associated with the given window.
syntax:
    [*colFrames*=]*object*.frames
    [*oObject*=]*object*.frames(*vIndex*)

colFrames:
    Collection of frames
oObject:
    Reference to an individual item in the array of elements contained by the object.
vIndex:
    Required. Integer that specifies the element or collection to retrieve.
    Note: *the value of vIndex cannot be accesed as a string.*

Attributes

none

Properties

length – read only property that contains the number of frames in the frames collection.

Methods, Events, Collections

none

<FRAMESET> element/object

Specifies a frameset consisting of 1 or more frames. The FRAMESET element is a block element and requires a closing tag.

Attributes

BORDER - iSpace: specifies the number of pixels to reserve as space between frames. If the value is <3, then Pocket IE uses the value of 3.
BORDERCOLOR - sColor: this attribute is not supported in Pocket IE.
COLS - sWidth: string consisting of comma separated values that specify the width of the FRAMEs. Values can be in pixels, percentage of the available width, or width*. In the case of width*, the width value is a relative value. After allocating space for pixel & percentage width sized frames, the remaining space is divided amongst relative-sized frames.
See http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/cols_2.asp for examples.
ROWS - sHeight: string consisting of comma separated values that specify the height of the FRAMEs. Values can be in pixels, percentage of the available width, or height*. In the case of height*, the height value is a relative value. After allocating space for pixel & percentage height sized frames, the remaining space is divided amongst relative-sized frames.
See http://msdn.microsoft.com/workshop/author/dhtml/reference/properties/rows_1.asp for examples.

Properties, Methods

none

Events

onload - event which fires when the frameset is loaded.
***Note:** The onload event will **not** fire if the user refreshes a page. For frames pages, the onload event does not fire until all frames in the frameset are loaded.*
onunload - fires immediately before the frameset is unloaded.

Collections

none

<Hn> (1...6) element

Renders text in heading style. Use H1 through H6 to specify different sizes and styles of headings. The Hn element is a block element and requires a closing tag.

Attributes

ALIGN - sAlign: specifies the alignment of the heading. Possible values are left, center, and right.

Properties, Methods, Events, Collections
none

<HEAD> element
Provides an unordered collection of information about the document. The HEAD element requires a closing tag.

Attributes, Properties, Methods, Events, Collections
none

history object
Contains information about the URLs visited by the client.

Attributes
none

Properties
length – iLength: read only property which always returns 0

Methods
back - syntax: back(): loads the previous URL from the history list. This is the same as pressing the back button.
forward - syntax: forward(): loads the next URL from the history list. This is the same as pressing the forward button.
go - syntax: go(iIndex): loads an URL from the history list. Valid values for iIndex are -1, 0, and 1.
Note: *go(0) does count as a navigation.*

Events, Collections
none

<HR> element
Draws a horizontal rule. The HR element is a block element and does not require a closing tag.

Attributes
ALIGN - sAlign: specifies the alignment of the HR. Possible values are center (default), left, and right.
NOSHADE: when specified indicates that the HR is not to be drawn w/3D shading.
SIZE - iSize: specifies the height of the HR in pixels.
WIDTH - iWidth: specifies the width of the HR in pixels or as a percentage width of its container.

Properties, Methods, Events, Collections
none

<HTML> element
Identifies the document as containing HTML elements. The HTML element requires a closing tag.

Attributes, Properties, Methods, Events, Collections
none

<I> element
Specifies that the enclosed text should be rendered in italics. The I element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections
none

<IMG> element
Embeds an image in the document.
Note: *IMG ALIGN and BORDER are not supported.*

Attributes
ALT - sText: specifies a text alternative to the graphic
HEIGHT - sHeight: specifies the height of the image in pixels or as a percentage of the parent object.
HSPACE - iMargin: sets the horizontal margin for the image in pixels.
ISMAP - specifies that the image is a server side image map
SRC - sSRC: specifies the URL of the image to be displayed. (Pocket IE supports .JPEG, .JPG, .GIF, .PNG, .2BP, .XBM, and .2BP files.)
USEMAP - sURL: specifies the URL of the client side image map (usually a bookmark like #map)
VSPACE - iMargin: sets the vertical margin for the image.
WIDTH - sWidth: specifies the width of the image in pixels or as a percentage of the parent object.

Properties, Methods, Events, Collections
none

<INPUT TYPE=button> element/object
Creates a button control. The input type=button element is an inline element and does not require a closing tag.

Attributes

NAME - sName: specifies the name of the button control.
VALUE - sValue: specifies the value of the control. This text is displayed as the button's label.
***Note:** buttons wider than the screen are visually clipped.*

Properties

type - sType: read only property that always returns: button
value - sValue: read/write property sets or retrieves the value of the input type=button.

Methods

blur - syntax: *button*.blur(): this method is not supported in Pocket IE.
click - syntax: *button*.click(): simulates clicking the button causing the onclick event to fire.
focus - syntax: *button*.focus(): sets focus to the button.

Events

onclick - fires when the user taps on the button element. Note the event only fires if the user does a pen down while on the button and a pen up over the same button.   The author can specify special handler for this event.

Collections

none

<INPUT TYPE=checkbox> element/object

Creates a checkbox control. The checkbox element is an inline element and does not require a closing tag.

Attributes

NAME - sName: specifies a name for the checkbox which is submitted with the VALUE if the checkbox is checked.
VALUE - sValue: specifies a VALUE for the checkbox which is submitted with the NAME of the checkbox if the checkbox is checked.
CHECKED - specifies that the checkbox is checked.

Properties

checked – bChecked: read/write property that sets or retrieves the state of the input type=checkbox control. The only possible values are true and false.
defaultChecked - bChecked: read only property that specifies whether the input type=checkbox is checked by default.
type - sType: read only property that always returns: checkbox
value - sValue: read/write property that sets or retrieves the value of the input type=checkbox.

Methods

blur - syntax: *checkbox*.blur(): this method is not supported in Pocket IE.
click - syntax: *checkbox*.click(): simulates a clicking on the checkbox by causing the onclick event to fire.
focus - syntax: *checkbox*.focus(): sets focus to the checkbox.

Events

onclick - fires when the user taps on the checkbox element. Note the event only fires if the user does a pen down while on the checkbox and a pen up over the same checkbox. The author can specify special handler for this event.

Collections

none

<INPUT TYPE=hidden> element/object

Used to transmit information about the client/server interaction. This control is not rendered but sends the name/value pair as defined by the attributes below. The hidden element does not require a closing tag.

Attributes

NAME - sName: specifies the name for the input type=hidden control. The NAME is submitted along with the associated VALUE during forms submission.
VALUE - sValue: specifies the value for the input type=hidden control. The VALUE is submitted along with the associated NAME during forms submission.

Properties

defaultValue - sDefault: read only property that retrieves the initial contents of the input type=hidden control
type - sType: read only property that always returns: hidden
value - sValue: read/write property that sets or retrieves the value of the input type=hidden control.

Methods

blur - syntax: *hidden*.blur(): this method is not supported in Pocket IE.

Events, Collections

none

<INPUT TYPE=image> element/object

Creates an image control, when clicked, causes the form to be immediately submitted. The coordinates where the click are measured from the upper-left of the image and are submitted w/the form as two name/value pairs. The x-coordinate is submitted under the name of the control with .x appended, and the y-coordinate is submitted under the name of the control with .y appended.

Attributes

NAME - sName: specifies the name for the input type=image control.
VALUE - sValue: specifies the value for the input type=value control.
SRC - sURL: specifies the URL of the image to be loaded by the input type=image.
HEIGHT - sHeight: specifies the height of the image in pixels or as a percentage of the parent object.
WIDTH - sWidth: specifies the width of the image in pixels or as a percentage of the parent object.

Properties

none

Methods

blur - syntax: *inputimage*.blur(): this method is not supported in Pocket IE.

Events, Collections

none

<INPUT TYPE=password> element/object

Creates a single-line text entry control similar to the text control, except that text is not displayed as the user enters it. The input type=password element is an inline element and does not require a closing tag.

Attributes

MAXLENGTH - sMaxlength: specifies the maximum number of characters a user can enter into the control.
NAME - sName: specifies the name for the input type=password control. The NAME is submitted along with the associated VALUE during forms submission.
VALUE - sValue: specifies the initial value for the input type=password control. The VALUE is submitted along with the associated NAME during forms submission.
SIZE - sSize: specifies the size of the input type=password in characters (roughly).
Note: *a size that would result in the control being too wide to fit on the screen of the device will be limited to slightly less than the screen width.*

Properties

defaultValue - sDefault: read only property which retrieves the default value (initial contents) of the input type=password control.
type - sType: read only property that always returns: password
value - sValue: read/write property that sets or retrieves the value of the input type=password control

Methods

blur - syntax: *password*.blur(): this method is not suported in Pocket IE.
focus - syntax: *password*.focus(): sets focus to the input type=password control.
select - syntax: *password*.select(): this method is not suported in Pocket IE.

Events

onchange - fires when the contents of the input type=password have changed and the user commits the change (by leaving the input type=password which has focus).

Collections

none

<INPUT TYPE=radio> element/object

Creates a radio button control. The input type=radio element is an inline element and does not require a closing tag. Use a radio button control to limit a user's selection to a single value within a set of values. To do this, you must group each button in a set of radio buttons by assigning each button the same name. When a user submits a form, a selected radio button only generates a name/value pair in the form data if the button has a value.
Note: *if a radio button is in a group and the button has no value, the button name is submitted without a value.*

Attributes

NAME - sName: specifies a name for the input type=radio control.
VALUE - sValue: specifies a value for the input type=radio control.
CHECKED - specifies that the input type=radio control is checked.

Properties

checked - bChecked: read/write property that sets or retrieves the state of the input type=radio control. The only possible values are true and false.
defaultChecked – bDefaultchecked: read only property that specifies whether or not the input type=radio is checked by default.
type - sType: read only property that always returns: radio
value - sValue: read/write property that sets or retrieves that value of the input type=radio control. The default value is “on”.

Methods

blur - syntax: *radio*.blur(): this method is not suported in Pocket IE.
click - syntax: *radio*.click(): simulates clicking on the input type=radio by causing the onclick event to fire.
focus - syntax: *radio*.focus(): sets focus to the input type=radio control.

Events

onclick - fires when the user taps on the radio button element. Note the event only fires if the user does a pen down while on the radio button and a pen up over the same radio. The author can specify special handler for this event.

Collections

none

<INPUT TYPE=reset> element/object

Creates a button that, when clicked, resets the form's controls to their initial values. The reset element is an inline element and does not require a closing tag.

Attributes

NAME - sName: specifies the name for the input type=reset control.
VALUE - sValue: specifies the value for the input type=radio control. The value is displayed as the label of the control.

Properties

type - sType: read only property that always returns: reset
value - sValue: read/write property that set or retrieves the value of the input type=reset button.

Methods

blur - syntax: *reset*.blur(): this method is not suported in Pocket IE.
click - syntax: *reset*.click(): simulates a click on the input type=reset by causing the onclick event to fire.
focus - syntax: *reset*.focus(): sets focus to the input type=reset control.

Events

onclick - fires when the user taps on the reset button element. Note the event only fires if the user does a pen down while on the reset button and a pen up over the same reset button.   The author can specify special handler for this event.

Collections

none

<INPUT TYPE=submit> element/object

Creates a button that, when clicked, submits the form. The input type=submit element is an inline element and does not require a closing tag.

Attributes

NAME - sName: specifies the name for the input type=submit control.
VALUE -sValue: specifies the value for the input type=submit control. If the input type=submit has a name and value and that user taps on the control, then the name/value pair is submitted to the server.

Properties

type - sType: read only property that always returns: submit
value - sValue: read/write property that sets or retrieves the value of the input type=submit control.

Methods

blur - syntax: *submit*.blur(): this method is not suported in Pocket IE.
click - syntax: *submit*.click(): simulates a click on the input type=submit by causing the onclick event to fire.
focus - syntax: *submit*.focus(): sets focus to the input type=submit control.

Events

onclick - fires when the user taps on the submit button element. Note the event only fires if the user does a pen down while on the submit button and a pen up over the same submit button. The author can specify special handler for this event.

Collections

none

<INPUT TYPE=text> element/object

Creates a single-line text entry control. The input type=text element is an inline element and does not require a closing tag.
***Note:** if the author places an INPUT without a TYPE attribute or with an invalid TYPE, Pocket IE treats it as an INPUT TYPE=text.*

Attributes

MAXLENGTH - iMaxlength: specifies the maximum length of the input type=text control in characters.
NAME - sName: specifies the name for the input type=text control.
VALUE - sValue: specifies the initial value for the input type=text control. If the control has a name and a value (either set by attribute, script or user input), on forms submission, the name/value pair is submitted to the server.
SIZE - iSize: specifies the size of the input type=text control in characters.

Properties

defaultValue - sDefaultvalue: read only property which retrieves the default value (initial contents) of the input type=text control.
type - sType: read only property that always returns: reset
value - sValue: read/write property that sets or retrieves the value of the input type=text control.

Methods

blur - syntax: text.blur(): this method is not suported in Pocket IE.
focus - syntax: text.focus(): sets focus to the input type=text control.
select - syntax: text.select():this method is not suported in Pocket IE.

Events

onchange - fires when the contents of the input type=text have changed and the user commits the change (by leaving the input type=text which has focus).

Collections
none

<ISINDEX> element
This tag is not supported in Pocket IE.

<KBD> element
Renders enclosed text in a fixed-width font. The KBD element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections
none

<LI> element
Denotes one item in a list. The LI element is an inline element and does not require a closing tag.

Attributes
TYPE - sType: specifies the style of the list. Possible values are:

circle	
square	
disc	
a	results in list items a, b, c, etc.
A	results in list items A, B, C, etc.
1	default value for list, results in list items 1, 2, 3, etc.
i	results in list items i, ii, iii, etc.
I	results in list items I, II, III, etc.

VALUE - sValue: sets the count that the current list item is at.
For example, setting a <LI VALUE="30"> would result in a the list items 30, 31, 32, etc.

Properties, Methods, Events, Collections
none

links collection
Retrieves a collection all of the <A> (link) objects which have a HREF and all AREA objects in the document.

Attributes
none

Properties
length – iLength: read only property that retrieves the number of elements in the links collection.

Methods, Events, Collections
none

location object
Object which contains information about the current URL.

Attributes
none

Properties
hash – sHash: read/write property that sets or retrieves the part of URL starting from the # inclusive and beyond. If you change the hash, the browser will scroll up/down to the appropriate bookmark.
Note: *changing the hash value will scroll the browser, however the value of the has property will not reflect this change.*
host – sHost: read/write property that sets or retrieves the hostname of the location of page.
Note: *in Pocket IE, this property will note return the port number.*
hostname – sHostname: read/write property that sets or retrieves the host name part of the location of the page.
href - sHref: read/write property that sets or retrieves the entire URL as a string
pathname - sPath: read/write property that sets or retrieves the file name or path of the location of the page.
port - iPort: read/write property that sets or retrieves the port number associated with the URL.
search - sSearch: read/write property that sets or retrieves the search (query) string portion of the URL. This includes the leading question mark.

Methods
reload - syntax: *location*.reload(): reloads the current page.
Note: *This method does not take the parameter bReloadSource, which specifies whether to check reload the page from the browser's cache or to reload it from the server. Pocket IE always checks against the server if the user is online.*
replace - syntax: *location*.replace(sURL) - replaces the current document by loading the document at the specified URL (in the required parameter sURL)

Events, Collections
none

<MAP> element
Defines a client side image map which contains one or more AREAs that specify hot zones on the associated image and bind these zones to URLs. The MAP element requires a closing tag.

Attributes

NAME - sName: specifies the name for the image map which is used to associate the map with an image. For example, for a MAP w/name SystemMap, you'd need to have an IMG USEMAP="#SystemMap".

Properties, Methods, Events, Collections

none

<MENU> element

Creates an unordered list of items (consisting of LI elements).

Attributes, Properties, Methods, Events, Collections

none

<META> element

Conveys hidden information to the server and the client. The META element does not require a closing tag.

Attributes

CONTENT - sContent: specifies the content to be associated with HTTP-EQUIV. Pocket IE looks at values of the format: iRefresh; [sURL]. iRefresh specifies the number of seconds before the document is refresh. sURL is optional and contains the URL of the document to be loaded on refresh

> *Example:*
> <META HTTP-EQUIV="REFRESH" CONTENT=2> causes the document to be refreshed every two seconds

HTTP-EQUIV - sInformation: specifies information used to bind the META tag's CONTENT to an HTTP response header.

Properties, Methods, Events, Collections

none

navigator object

Object which contains information about the browser.

Attributes

none

Properties

appCodeName – sCodeName: read only property that retrieves the code name of the browser. The value is always "Mozilla".
appName – sName: read only property that retrieves the name of the browser. The value is always "Microsoft Internet Explorer".

appVersion – sVersion: read only property that retrieves the platform and version of the browser. The value is always 2.0 (compatible: MSIE 3.02; Windows CE).
userAgent – sAgent: read only property that retrieves a string equivalent to the HTTP user-agent request header. The value is always Mozilla/2.0 (compatible; MSIE 3.02; Windows CE; 240x320).

Methods

javaEnabled - syntax: bEnabled=javaEnabled() - returns whether Java is enabled. This always returns false on Pocket IE.
taintEnabled - syntax: bEnabled=taintEnabled() - returns whether data tainting is enabled. This always returns false on Pocket IE.

Events, Collections

none

<NOFRAMES> element

Contains HTML for browsers that do not support FRAMES. The NOFRAMES element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<NOSCRIPT> element

Specifies HTML to be displayed in browsers which do not support scripting. The NOSCRIPT element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<OBJECT> element/object

Inserts an ActiveX control onto the page. The OBJECT element is a block element and requires a closing tag.
Note: *There is no mechanism in Pocket IE to download controls. The author is limited to using controls that have been pre-installed on the device.*

Attributes

ALIGN - sAlign: specifies the alignment for the control. Possible values are:

baseline
top
center
middle
bottom

CLASS - sClass: not supported in Pocket IE.

CLASSID - sClassId: specifies the class identifier for the control. The format is "clsid:XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX". This attribute is mandatory for the control to render.
HEIGHT - sHeight: specifies the height of the ActiveX control in pixels
HSPACE - iMargin: sets the horizontal margin for the control in pixels.
ID - sId: specifies the string identifying the object. Can be used so that the OBJECT can be referenced from script. This attribute is mandatory for the control to render.
NAME - sName: not supported in Pocket IE.
VSPACE - iMargin: sets the vertical margin for the control in pixels.
WIDTH - sWidth: specifies the width of the ActiveX control in pixels

Properties, Methods, Events, Collections

none

<OL> element

Creates a numbered list consisting of LI elements.

Attributes

START - iStart: specifies the starting number for the ordered list
TYPE - sType: specifies the style of the list. Possible values are:

a	results in list items a, b, c, etc.
A	results in list items A, B, C, etc.
1	results in list items 1, 2, 3, etc.
i	results in list items i, ii, iii, etc.
I	results in list items I, II, III, etc.

Properties, Methods, Events, Collections

none

<OPTION> element/object

Denotes one choice in a SELECT element. The OPTION element does not require a closing tag.

Attributes

SELECTED - specifies that the option is the selected
VALUE - sValue: specifies the value of the option. This value is submitted if the option is selected.

Properties

defaultSelected - bdefaultSelected: read only property which retrieves whether the option is selected by default (via the presence of the SELECTED attribute in the OPTION tag). The only possible values are true and false.
index - iIndex: read only property which retrieves the zero-based index of the OPTION in the options collection of the SELECT

selected – bSelected: read/write property that sets or retrieves whether the OPTION is selected. The only possible values are true and false.
text – sText: read/write property which sets or retrieves the text string specified by the OPTION tag.
value - sValue: read/write property which sets or retrieves the value of the OPTION.

Methods, Events, Collections

none

options collection

Retrieves a collection of OPTION objects in a SELECT object.
Syntax:

[*colOptions*=]*select*.options
[*oObject*=]*select*.options(*vIndex*)

colOptions:

Collection of options

oObject:

Reference to an individual item in the array of elements contained by the object.

vIndex:

Required. Integer that specifies the element to retrieve.
Note: *the value of vIndex cannot be accesed as a string.*

Attributes

none

Properties

length - iLength: read only property which retrieves the number of options in the collection.

Methods, Events, Collections

none

<P> element

Denotes a paragraph. The P element is a block element and does not require a closing tag.

Attributes

ALIGN - sAlign: specifies the alignment of the paragraph. Possible values are left, center, and right.

Properties, Methods, Events, Collections

none

<PARAM> element

Sets the property value for a given OBJECT. This tag must be inside the associated OBJECT tag. The PARAM element does not require a closing tag.

Attributes

NAME - sName: specifies the name of the parameter passed to the OBJECT
VALUE - sValue: specifies the value of the parameter passed to the OBJECT

Properties, Methods, Events, Collections

none

<PRE> element

Denotes preformatted text and renders it in a fixed pitch font. Spaces and carriage returns within the PRE tag are preserved. The PRE element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<Q> element

This tag is not supported in Pocket IE.

<S> element

Renders text in strike-through type. The S element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<SAMP> element

Denotes a code sample. Contents inside the tag are rendered in a fixed pitch font. The SAMP element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<SCRIPT> element

Specifies a block containing script to be interpreted by the script engine. (Note: external script files pointed to by SCRIPT SRC are not supported by Pocket IE)

Attributes

EVENT - sEvent: specifies the name of the event handler that the enclosed script is bound to.
Note: *the only supported event is onload.*

FOR - sFor: specifies the name of the object that the event script is bound to.
Note: *the only supported objects are window and document.*

Example:
```
<html>
  <head>
    <script for=window event="onload">
      alert('onload on window fired')
    </script>
  </head>
</html>
```

LANGUAGE - sLanguage: specifies the language in which the script is written. The default value if the attribute or value is omitted is JScript. The following are values are equivalent: jscript, javascript, javascript1.1. If any other value is specified (e.g. javascript1.2, vbs, vbscript, etc.), the script is not parsed and is not executed.

Properties, Methods, Events, Collections
none

<SELECT> element/object
Denotes a list box or drop-down list. The SELECT element is an inline element and requires a closing tag.

Attributes
MULTIPLE - specifies that multiple items can be selected
NAME - sName: specifies a name for the SELECT which is submitted (a name/value pair is submitted for each selected OPTION) and can be used to refer to the control via script.
SIZE - iSize: specifies the size for the SELECT control. The default value is 1, which results in a combo box being rendered. For values greater than 1, a list box is rendered.

Properties
selectedIndex - read/write property that sets or retrieves the index of the selected option in a SELECT object. This index is 0 based and returns -1 if no items are selected. Setting the selectedIndex clears any existing selected items. Note, this property is not very useful for SELECT w/the MULTIPLE attribute, as it only returns the index of the first selected option.
type - read only property that always returns:
select-one for SELECT controls without the MULTIPLE attribute
select-multiple for SELECT controls with the MULTIPLE attribute
value - this property is **not** supported in Pocket IE.

Methods
blur - syntax: *select*.blur(): this property is **not** supported in Pocket IE.
focus - syntax: *select*.focus(): sets focus to the SELECT control.

Events

onchange - fires when the user changes the selection in the SELECT control. It does not fire when the selection is changed programmatically.

Collections

options - Collection of all the <OPTION> elements/objects in the SELECT. See options collection for more information.

<SMALL> element

Specifies that the enclosed text should be rendered in a smaller font.

Attributes, Properties, Methods, Events, Collections

none

<STRIKE> element

Renders enclosed text in strike-through type. The STRIKE element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<STRONG> element

Renders enclosed text in boldface. The STRONG element is an inline element and requires a closing tag

Attributes, Properties, Methods, Events, Collections

none

<STYLE> element

This tag is not supported in Pocket IE. The contents of this tag are ingored.

<SUB> element

Specifies that the enclosed text should be displayed in subscript, using a smaller font than the current font. The SUB element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<SUP> element

Specifies that the enclosed text should be displayed in superscript, using a smaller font than the current font. The SUP element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<TABLE> element

Specifies that the contained content is organized into a table with rows and columns. The TABLE element is a block element and requires a closing tag.

Attributes

BGCOLOR - sColor: sets the background color of the table. See BODY BGCOLOR for the format and possible color names. Note: Table background colors are intentionally disabled on grayscale devices resulting in table backgrounds always being white.
BORDER - iBorder: sets the width of the border in pixels to be drawn around the table and in between cells. Omitting the attribute or setting it to zero will cause no borders to be drawn on the table.
BORDERCOLOR - sColor: sets the border color of the table. See BODY BGCOLOR for the format and possible color names. Note: Table border colors are intentionally disabled on grayscale devices resulting in table border colors always being black.
CELLPADDING - iPadding: specifies the amount of space in pixels between the border of the cell and the content of the cell (percentage values are not supported)
CELLSPACING - iSpacing: specifies the amount of space in pixels between table cells (percentage values are not supported)
FRAME - sFrame: specifies the way the border frame around the table is displayed. Possible values are:

void:	all outside table borders are removed
above:	border on the top side of the border frame is displayed
below:	border on the bottom side of the border frame is displayed
hsides:	borders on the top and bottom sides of the table frame are displayed
lhs:	border on the left side of the table frame is displayed
rhs:	border on the right side of the table frame is displayed
vsides:	borders on the left and right sides of the table frame are displayed
box:	borders on all sides of the table frame are displayed
border:	borders on all sides of the table frame are displayed

RULES - (See comments for TABLE FRAME attribute) sRules: Specifies how rules (inner borders) are displayed. Possible values are:

none:	all interior table borders are removed
rows:	horizontal borders are displayed between all table rows
cols:	borders are displayed between all table columns
all:	borders are displayed on all rows and columns

WIDTH - sWidth: specifies the width of the table in pixels or as a percentage of the parent object.

Note: *TABLE ALIGN= is not supported.*

Properties, Methods, Events, Collections

none

<TBODY> element

This tag does nothing in Pocket IE. Designates rows as the body of the table. The TBODY element is a block element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<TD> element

Specifies a cell in a table. The TD element is a block element and requires a closing tag.

Attributes

ALIGN - sAlign: specfies the alignment of the table cell. Possible values are left (default), center, and right.
BGCOLOR - sColor: sets the background color of the table cell. See BODY BGCOLOR for the format and possible color names. Note: Table cell background colors are intentionally disabled on grayscale devices resulting in table backgrounds always being white.
COLSPAN - iCount: sets the number of columns in the table that the table cell should span
ROWSPAN - iCount: sets the number of rows in the table that the table cell should span
VALIGN -sAlign: specifies the vertical alignment of the contents of the table cell. Possible values are:

baseline:	aligns the base line of the first line of text with the base lines in adjacent table cells.
top:	aligns contents with the top of the table cell
center:	aligns contents with the middle of the table cell
middle:	aligns contents with the middle of the table cell
bottom:	aligns contents with the bottom of the table cell

WIDTH - sWidth: specifies the width of the table cell in pixels.
Note: *Percentage widths are not supported.*

Note: *TD HEIGHT is not supported.*

Properties, Methods, Events, Collections

none

<TEXTAREA> element/object

Specifies a multi-line text input control. The TEXTAREA element is an inline element and requires a closing tag.
Note: *the WRAP attribute is not supported by Pocket IE.*

Attributes

COLS - iCols: specifies the number of columns in the TEXTAREA which is used to determine its width. The default value of this attribute is 50.
NAME - sName: specifies the name of the TEXTAREA which is submitted with its contents
ROWS - iRows: specifies the number of rows in the TEXTAREA which is used to determine its height. The default value of this attribute is 5.

Properties

defaultValue - read only property which retrieves the default value (initial contents) of the textarea control.
type - read only property that always returns: textarea
value - read/write property that sets or retrieves the value of the textarea control.

Methods

blur - syntax: *textarea*.blur(): is supposed to cause the textarea to lose focus and fire the onblur event. However, in Pocket IE, this call does nothing and does not return an error.
focus - syntax: *textarea*.focus(): sets focus to the textarea control.
select - syntax: *textarea*.select(): this method is not supported in Pocket IE.

Events

onchange - fires when the contents of the input type=text have changed and the user commits the change (by leaving the textarea which has focus).

Collections

none

<TFOOT> element

This tag is not supported in Pocket IE.

<TH> element

Specifies a header column of a table. Contents of the TH are centered within the cell and bold, unless overridden. The TH element is a block element and requires a closing tag.
Note: *this is almost the same as a TD except for the default formatting.*

Attributes

ALIGN - sAlign: specfies the alignment of the table header. Possible values are left, center (default), and right.
BGCOLOR - sColor: sets the background color of the table header. See BODY BGCOLOR for the format and possible color names.
Note: *To improve readability, table header background colors are disabled on grayscale devices resulting in table backgrounds always being white.*
COLSPAN - iCount: sets the number of columns in the table that the table header should span

ROWSPAN - iCount: sets the number of rows in the table that the table header should span
VALIGN -sAlign: specifies the vertical alignment of the contents of the table header. Possible values are:

baseline:	aligns the base line of the first line of text with the base lines in adjacent table cells.
top:	aligns contents with the top of the table cell
center:	aligns contents with the middle of the table cell
middle:	aligns contents with the middle of the table cell
bottom:	aligns contents with the bottom of the table cell

WIDTH - sWidth: specifies the width of the table header in pixels.
Note: *Percentage widths are not supported.*

Properties, Methods, Events, Collections

none

<THEAD> element

This tag is not supported in Pocket IE.

<TITLE> element

Contains the title of the document. The TITLE element is a block element and requires a closing tag. The title of the page is displayed when the user goes to View->Properties.

Attributes, Properties, Methods, Events, Collections

none

<TR> element

Specifies a row in a table. The TR element is a block element and requires a closing tag.

Attributes

ALIGN - sAlign : specifies the horizontal alignment of the table row. Possible values are:

left:	(default) aligns the row with the left edge of the table
center:	*aligns the row to the center of the table*
middle:	aligns the row to the center of the table
right:	aligns the row with the right edge of the table

BGCOLOR - sColor: sets the background color of the table row. See BODY BGCOLOR for the format and possible color names.
Note: *To improve readability, table row background colors are disabled on grayscale devices resulting in table backgrounds always being white.*
VALIGN -sAlign: specifies the vertical alignment of the contents of the table row. Possible values are:

baseline:	aligns the base line of the first line of text with the base lines in the adjacent table rows
top:	aligns contents with the top of the table row
center:	aligns contents in the middle of the table row
middle:	aligns contents in the middle of the table row
bottom:	aligns contents with the bottom of the table row

Properties, Methods, Events, Collections

none

<TT> element

Renders text in a fixed pitch font. The TT element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<U> element

Renders text that is underlined. The U element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<UL> element

Creates a bulleted list consisting of list items (LI). The UL element is a block element and requires a closing tag.

Attributes

PLAIN - specifies that the list be rendered without bullets.
TYPE - sType: specifies the style of the bullets in list. Possible values are
circle
square
disc (default)

Properties, Methods, Events, Collections

none

<VAR> element

Renders enclosed text in italics. The VAR element is an inline element and requires a closing tag.

Attributes, Properties, Methods, Events, Collections

none

<XML> element

Defines a XML data island.

Attributes

ID - sId: specifies so that the data island can be referenced from script.
SRC - sSrc: specifies the URL of an external XML file
***Note:** in Pocket IE, XML data is always loaded synchronously; the ASYNCH attribute is not supported*

Properties, Methods, Events, Collections

none

window object

Represents a window in the browser.

Attributes

none

Properties

history - oHistory: read only property/object which contains information about the URLs visited by the client. See history object for more information.
length – iLength: read only property which retrieves the number of frames in the window.
location – oLocation: this property is not supported in Pocket IE.
navigator - oNavigator: read only property/object which contains information about the web browser. See navigator object for more information.
parent – oParent: read only property that retrieves the parent of the window in the object hierarchy.
self – oSelf: read only property which retrieves a reference to the current window or frame.
top – oTop: read only property which retrieves the topmost ancestor window which is its own parent.

Methods

alert - syntax: *window*.alert(sMessage): displays a dialog containing the contents of the paramter sMessage, which is a required parameter.
blur - syntax: *window*.blur(): causes the browser window (Pocket IE) to lose focus, switching you to the previous application.
clearTimeout - syntax: *window*.clearTimeout(*iTimerID*): cancels a timeout that was set with the setTimeout method. The parameter iTimeoutID is required and should be set to the timeout id that was returned from setTimeout.
close - syntax: *window*.close(): is supposed to close the current browser window, but in Pocket IE, this call does nothing and does not cause an error.
confirm - syntax: bChoice=*window*.confirm(*sMessage*): displays a confirmation dialog that contains the message contained in sMessage as well as OK and Cancel buttons. The

parameter sMessage is required. The method returns true if the user clicks OK or false if the user clicks Cancel.
focus - syntax: *window*.focus(): sets focus to the window.
navigate - syntax: window.navigate(*sURL*): navigates the browser to the URL specified by sURL. The parameter sURL is required.
open - Pocket IE only supports a single browser window; as a result, this method is not supported.
prompt - syntax: *vTextData*=*window*.prompt(*sMessage* [, *sDefaultValue*]): displays a dialog box prompt the user with a message contained in sMessage and a text field. The text entered in the field is returned in vTextData. sMessage contains the message to be displayed and is required. sDefaultValue can contain an optional default value in the text box.
scroll - this method is not supported in Pocket IE.
setTimeout - syntax: *iTimerID*=*window*.setTimeout(*sCode*, *iMilliSeconds*): executed code specified by sCode after iMilliseconds have passed and returns iTimerID (which is used with clearTimeout). sCode must be a string and cannot be a function pointer.

Events

none

Collections:

frames - Retrieves a collection of all window objects defined by a given document or defined by the document associated with the given window. See frames collection for more information.