# HOL P101 - Creating a Hello World Padarn Web Page

## Table of Contents

Table of Contents .......... 2

HOL Requirements .......... 3

Summary .......... 4

Lab Objective .......... 4

Exercise 1:  Updating the Padarn Hosting Executable .......... 5

Exercise 2:  Creating the Page ASPX file .......... 10

Exercise 3: Creating the Page Code-Behind Class .......... 12

Exercise 4: Deploying the Page Files .......... 15

Exercise 5: Debugging the Page Code-Behind Assembly .......... 18

Hands-on Lab Summary .......... 21

## HOL Requirements

The following items are required to run this HOL:

- A Desktop PC running Microsoft Windows® XP or Windows Vista
- Microsoft Visual Studio 2008 Professional (or higher)
- A Web Browser
- A Padarn reference system or developer kit
- The completed Solution from lab HOL 100

While Padarn will run on almost any hardware that supports the Microsoft .NET Compact Framework 2.0 or higher, this lab assumes that you have one of the Padarn reference hardware platforms with an OpenNETCF-validated Windows CE image running on it. If you are using an alternate hardware or software configuration, the steps outlined in this Hands-on Lab may not be accurate for your environment.

If you are working on a platform other than a Padarn reference platform, a common problem is a contention between Padarn and the built-in Windows CE HTTP server for Port 80. If your device has a display and a Console built in you can stop the built-in server by typing the following in a Console window:

```
services stop htp0:
```

If your device does not have both a display and the Console window, see the MSDN documentation for information on stopping or disabling the built-in server.

## Summary

In this lab, you will learn how to create, deploy and debug a simple “Hello World” Web Page served up by OpenNETCF’s Padarn Web Server for Windows CE.

## Lab Objective

Upon completion of this lab, you will be able to create a Visual Studio 2008 Solution containing all of the necessary Projects to easily develop, deploy and debug a full Padarn web solution.

In this HOL, you will perform the following exercises:

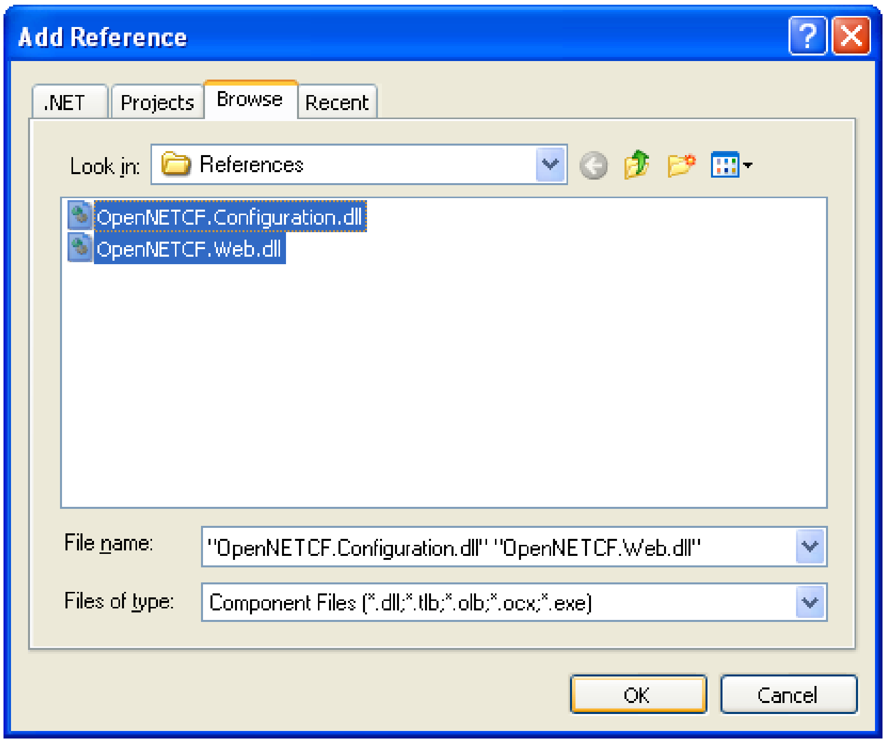
- ✓ Update the ocfhttpd.exe project to create an instance of the Padarn server and start it.
- ✓ Create the ASPX file for your Web Page
- ✓ Create the class containing the logic for your page in the code-behind assembly
- ✓ Deploy all files to your target device
- ✓ Debug your page code using the debugging tools in Visual Studio 2008

## Exercise 1: Updating the Padarn Hosting Executable

In this exercise, you will update the ocfhttpd.exe project code to create an instance of the Padarn WebServer class and use Form elements to affect its state.

The first task for updating the Padarn hosting executable will be to add a reference to the Padarn server assemblies. To add a Reference to the server assemblies:

1. Open the **MyPadarnSolution** Solution Microsoft Visual Studio 2008
2. Select the **ocfhttpd** Project in Visual Studio's **Solution Explorer** Pane
3. From Visual Studio's menu, select **Project -> Add Reference** to display the **Add Reference** dialog.
4. Select the **Browse** tab.
5. Browse to the location of the Padarn Assembly **OpenNETCF.Web.dll**. By default this assembly will be located at `c:\Program Files\OpenNETCF Consulting, LLC\OpenNETCF Padarn Web Server`, though in the code supplied with this HOL they are also included in the `References` folder.
6. Select both the **OpenNETCF.Web.dll** and the **OpenNETCF.Configuration.dll** Assemblies.
7. Click **OK** to add the References to the Project.



OpenNETCF.Configuration.dll
OpenNETCF.Web.dll

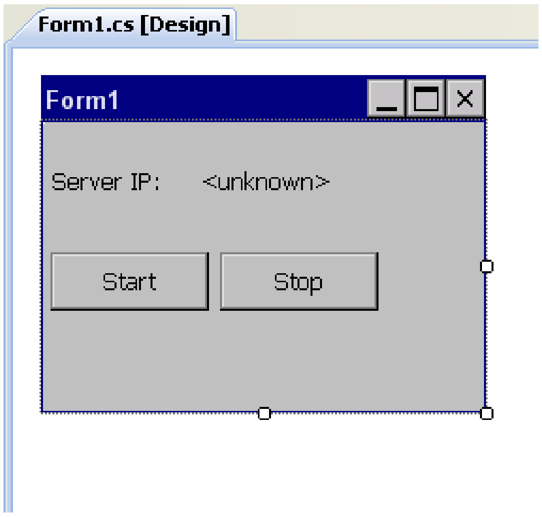
File name: "OpenNETCF.Configuration.dll" "OpenNETCF.Web.dll"

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe)

The next step in updating the hosting executable is to create a user interface (UI) to allow you to start and stop the web server. We will also have the UI display the server's IP address to make it easier to locate this information later.

To update the main Form UI of Padarn hosting executable:

1. In Visual Studio's Solution Explorer Pane, double-click on **Form1.cs** in the **ocfhttpd** Project to open Form1 in **Design Mode**.
2. **Delete** the **MainMenu1** component from the designer.
3. Drag two (2) **Label** controls from the **Toolbox** onto **Form1**.
4. Change the **Text** Property of **Label1** to '**Server IP:**'.
5. Change the **Name** Property of **Label2** to '**ipLabel**'.
6. Change the **Text** Property of **ipLabel** to '**<unknown>**'.
7. Drag two (2) **Button** controls from the **Toolbox** onto **Form1**.
8. Change the **Name** Property of **Button1** to '**startButton**'.
9. Change the **Text** Property of **startButton** to '**Start**'.
10. Change the **Name** Property of **Button2** to '**stopButton**'.
11. Change the **Text** Property of **stopButton** to '**Stop**'.



Next we must add the code for the logic behind the user interface.

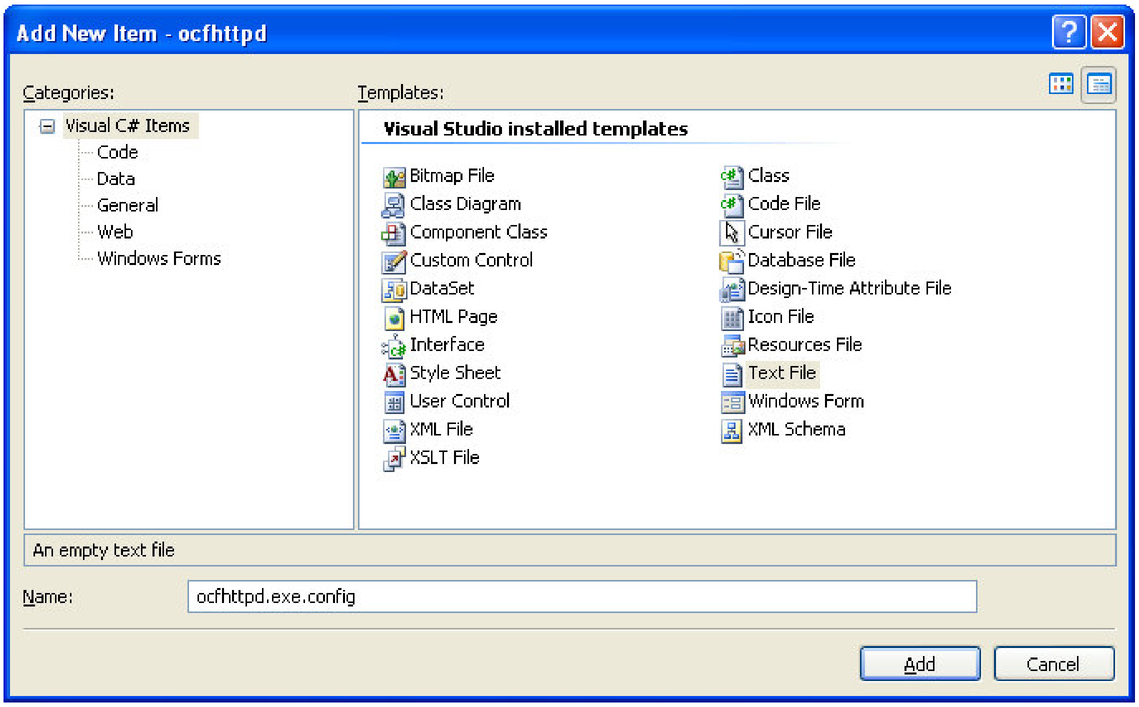
1. In Visual Studio's Solution Explorer Pane, right-click on **Form1.cs** in the **ocfhttpd** Project and select **View Code** to open the code page for Form1.
2. Update the code to match the following:

```
using System;
using System.Windows.Forms;
using OpenNETCF.Web.Server;
using System.Net;
using System.Net.Sockets;

namespace ocfhttpd
{
  public partial class Form1 : Form
  {
    private WebServer m_padarnServer = new WebServer();

    public Form1()
    {
      InitializeComponent();
```

```
          startButton.Click += new EventHandler(startButton_Click);
          stopButton.Click += new EventHandler(stopButton_Click);

          IPHostEntry localHost = Dns.GetHostEntry(Dns.GetHostName());
          for (int x = 0; x < localHost.AddressList.Length; x++)
          {
            if (localHost.AddressList[x].AddressFamily == AddressFamily.InterNetwork)
            {
              ipLabel.Text = localHost.AddressList[x].ToString();
              break;
            }
          }
        }

        void stopButton_Click(object sender, EventArgs e)
        {
          if (m_padarnServer.Running)
          {
            m_padarnServer.Stop();
          }
        }

        void startButton_Click(object sender, EventArgs e)
        {
          if (!m_padarnServer.Running)
          {
            try
            {
              m_padarnServer.Start();
            }
            catch (SocketException)
            {
              MessageBox.Show(
                "Port 80 is already in use.  Ensure no other web server is running.");
            }
          }
        }
      }
    }
```

Before we can run the Padarn host, we must provide configuration information for the Padarn server. To be an unobtrusive as possible, Padarn only requires a simple XML-based application configuration (app.config) file. To create the app.config file:

1. In Visual Studio's Solution Explorer Pane, right-click on the **ocfhttpd** Project and select **Add -> New Item** to display the **Add New Item** dialog.
2. From the list of **Visual Studio Installed Templates** select **Text File**.
3. In the **Name** textbox enter '**oncfhttpd.exe.config**'.



Add New Item - ocfhttpd

Categories:
- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - Windows Forms

Templates:

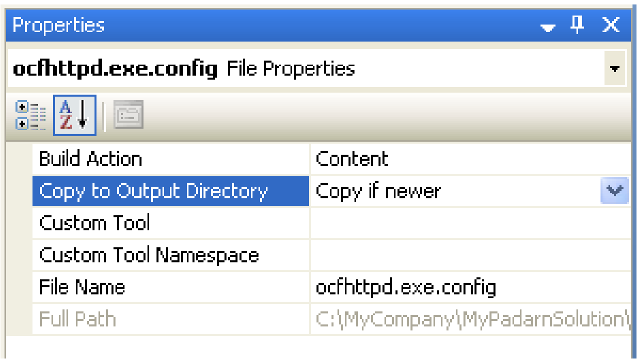
Visual Studio installed templates

| | |
|---|---|
| Bitmap File | Class |
| Class Diagram | Code File |
| Component Class | Cursor File |
| Custom Control | Database File |
| DataSet | Design-Time Attribute File |
| HTML Page | Icon File |
| Interface | Resources File |
| Style Sheet | Text File |
| User Control | Windows Form |
| XML File | XML Schema |
| XSLT File | |

An empty text file

Name: ocfhttpd.exe.config

4. Click Add to add the new file to the project.
5. In Visual Studio's Solution Explorer Pane, click on the newly-created **ocfhttpd.exe.config** file.
6. In the **Properties** pane, set the **Build Action** for **ocfhttpd.exe.config** to **Content**.
7. In the **Properties** pane, set the **Copy to Output Directory** property for **ocfhttpd.exe.config** to **Copy if Newer**.



Properties

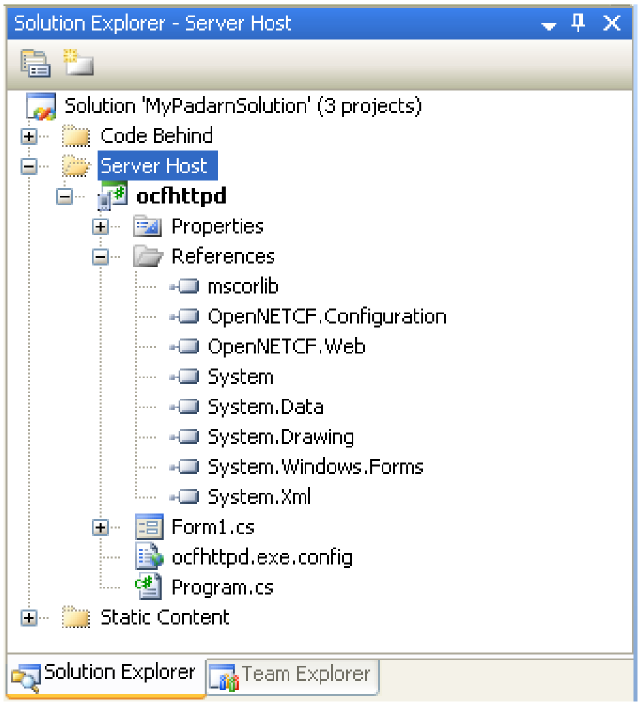
**ocfhttpd.exe.config** File Properties

| | |
|---|---|
| Build Action | Content |
| Copy to Output Directory | Copy if newer |
| Custom Tool | |
| Custom Tool Namespace | |
| File Name | ocfhttpd.exe.config |
| Full Path | C:\MyCompany\MyPadarnSolution\ |

8. Paste the following text into the newly-created ocfhttpd.exe.config file:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <section name="WebServer"

      type="OpenNETCF.Web.Configuration.ServerConfigurationHandler, OpenNETCF.Web" />
    <section name ="httpRuntime"
      type ="OpenNETCF.Web.Configuration.HttpRuntimeConfigurationHandler, OpenNETCF.Web"/>
  </configSections>

  <WebServer
    DefaultPort="80"
    MaxConnections="20"
    DocumentRoot="\Inetpub\"
    Logging="false"
  >
    <DefaultDocuments>
      <Document>default.aspx</Document>
    </DefaultDocuments>

    <VirtualDirectories />
  </WebServer>

  <httpRuntime
    maxRequestLength="4096"
    requestLengthDiskThreshold="256"
  />
</configuration>
```

9. The Solution will now look like this:



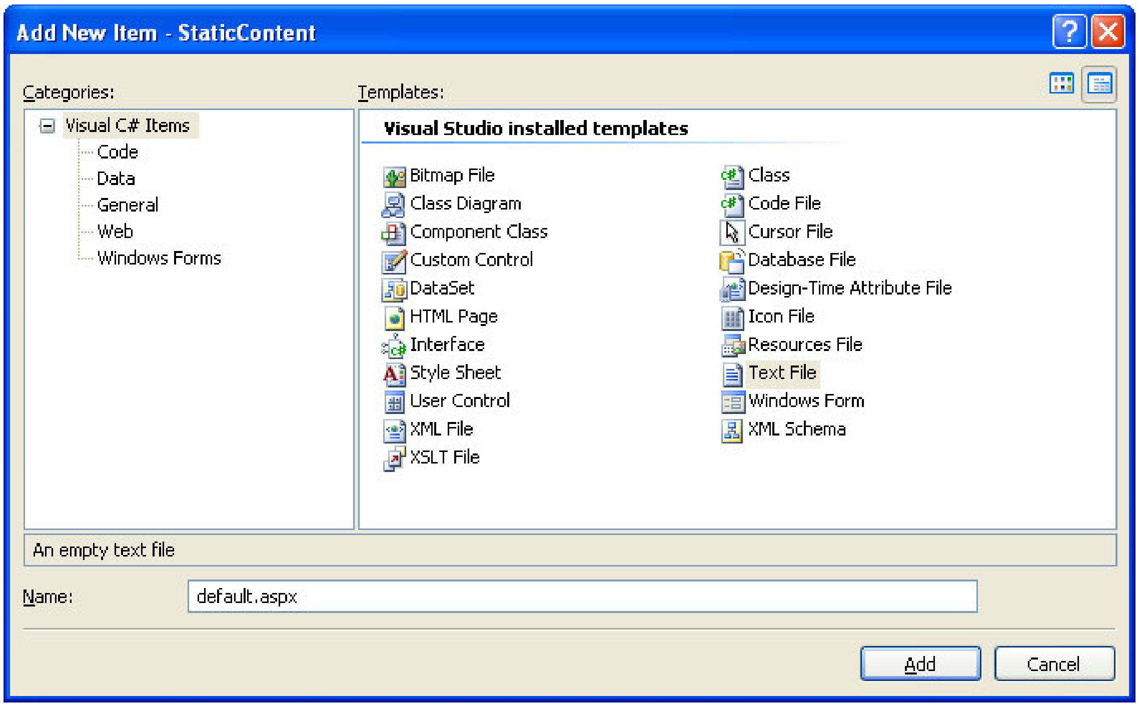
```
Solution 'MyPadarnSolution' (3 projects)
+ Code Behind
- Server Host
  - ocfhttpd
    + Properties
    - References
        mscorlib
        OpenNETCF.Configuration
        OpenNETCF.Web
        System
        System.Data
        System.Drawing
        System.Windows.Forms
        System.Xml
    + Form1.cs
      ocfhttpd.exe.config
      Program.cs
+ Static Content
```

## Exercise 2: Creating the Page ASPX file

In this exercise, you will use Visual Studio to create the ASPX file that describes to the Padarn server the code-behind assembly and class that it should load when a client browses to the page.

To create the ASPX file:

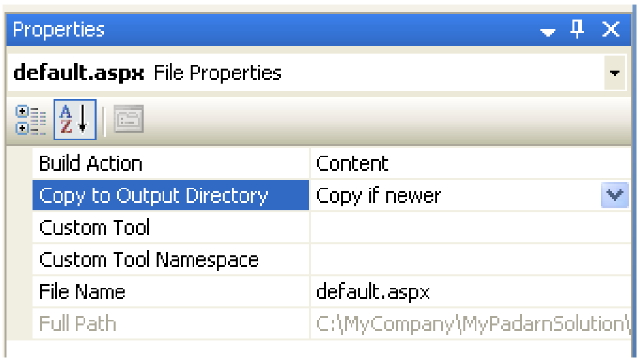
1. In Visual Studio's **Solution Explorer** Pane, right-click on the **Inetpub** folder in the **StaticContent** project and select **Add -> New Item** to display the **Add New Item** dialog.
2. From the list of **Visual Studio Installed Templates** select **Text File**.
3. In the **Name** textbox enter '**default.aspx**'.



An empty text file

Name: default.aspx

4. Click Add to add the new file to the project.
5. In Visual Studio's Solution Explorer Pane, click on the newly-created **default.aspx** file.
6. In the **Properties** pane, set the **Build Action** for **ocfhttpd.exe.config** to **Content**.
7. In the **Properties** pane, set the **Copy to Output Directory** property for **ocfhttpd.exe.config** to **Copy if Newer**.


Properties

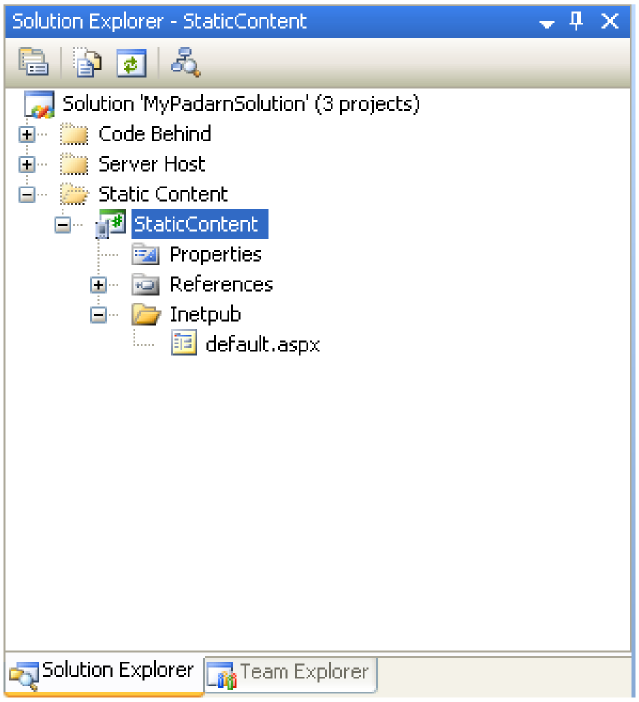
**default.aspx** File Properties

| | |
|---|---|
| Build Action | Content |
| Copy to Output Directory | Copy if newer |
| Custom Tool | |
| Custom Tool Namespace | |
| File Name | default.aspx |
| Full Path | C:\MyCompany\MyPadarnSolution\ |

8. Paste the following text into the newly-created **default.aspx** file:

```
<%@ Page CodeBehind="MyPageLibrary.dll" Inherits="MyPageLibrary.Default" %>
```

9. The Solution will now look like this:


Solution Explorer - StaticContent

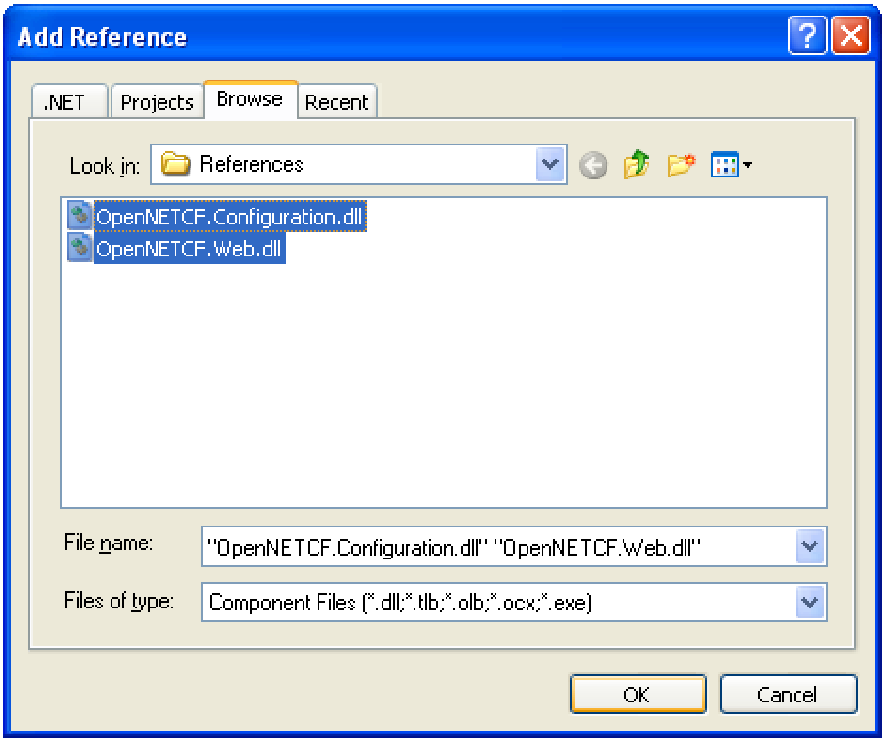
```
Solution 'MyPadarnSolution' (3 projects)
  Code Behind
  Server Host
  Static Content
    StaticContent
      Properties
      References
      Inetpub
        default.aspx
```

## Exercise 3: Creating the Page Code-Behind Class

In this exercise, you will use Visual Studio to a class containing the code-behind logic for your page. The Padarn server will create an instance of this class and run the `Page_Load` method of the class when a client browser navigates to your page.

The first task for creating the code-behind will be to add a reference to the Padarn server assemblies. To add a Reference to the server assemblies:

1. Open the **MyPadarnSolution** Solution Microsoft Visual Studio 2008
2. Select the **ocfhttpd** Project in Visual Studio's **Solution Explorer** Pane
3. From Visual Studio's menu, select **Project -> Add Reference** to display the **Add Reference** dialog.
4. Select the **Browse** tab.
5. Browse to the location of the Padarn Assembly **OpenNETCF.Web.dll**. By default this assembly will be located at `c:\Program Files\OpenNETCF Consulting, LLC\OpenNETCF Padarn Web Server`, though in the code supplied with this HOL they are also included in the `References` folder.
6. Select both the **OpenNETCF.Web.dll** and the **OpenNETCF.Configuration.dll** Assemblies.
7. Click **OK** to add the References to the Project.



OpenNETCF.Configuration.dll
OpenNETCF.Web.dll

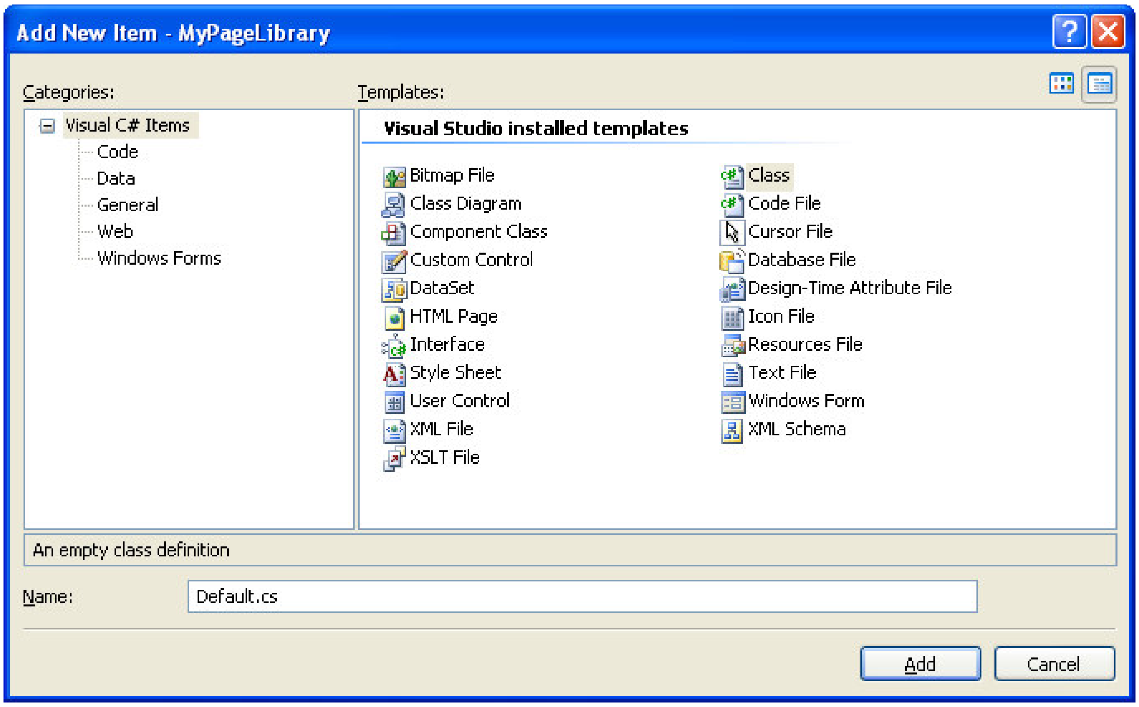
File name: "OpenNETCF.Configuration.dll" "OpenNETCF.Web.dll"

Files of type: Component Files (*.dll;*.tlb;*.olb;*.ocx;*.exe)

The next step in creating the code-behind assembly is to create the class that contains the page logic.

To create the code-behind class:

1. In Visual Studio's **Solution Explorer** Pane, right-click on the **MyPageLibrary** project and select **Add -> New Item** to display the **Add New Item** dialog.
2. From the list of **Visual Studio Installed Templates** select **Class**.
3. In the **Name** textbox enter '**Default.cs**'.



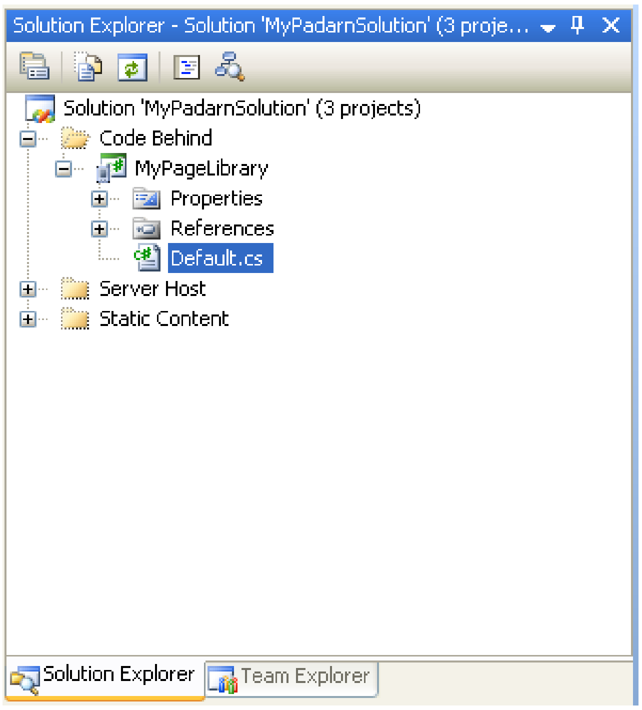
An empty class definition

Name: Default.cs

4. Click Add to add the new file to the project.
5. Change the newly-created **Default.cs** file to look like the following:

```
using System;
using OpenNETCF.Web.UI;

namespace MyPageLibrary
{
  public class Default : Page
  {
    protected override void Page_Load(object sender, EventArgs e)
    {
      Response.WriteLine("<html><body><h1>Hello World!</h1></body></html>");
      Response.Flush();
    }
  }
}
```

6. The Solution will now look like this:

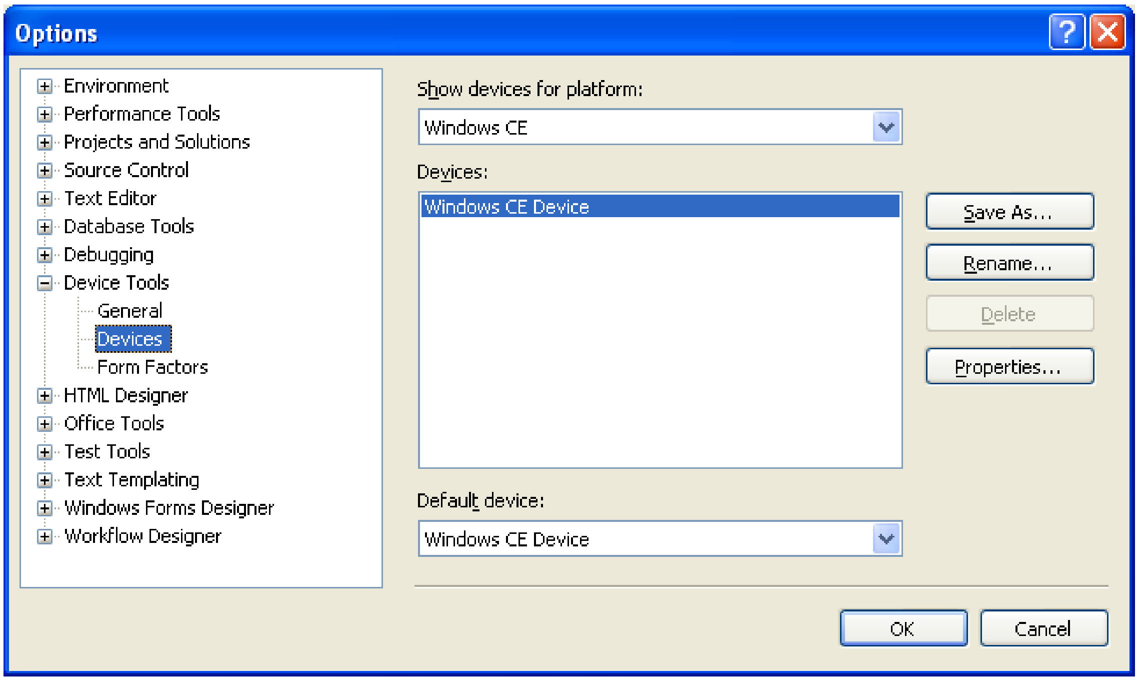

Solution Explorer - Solution 'MyPadarnSolution' (3 proje...
Solution 'MyPadarnSolution' (3 projects)
Code Behind
MyPageLibrary
Properties
References
Default.cs
Server Host
Static Content
Solution Explorer
Team Explorer

## Exercise 4: Deploying the Page Files

In this exercise you will connect Visual Studio 2008 to your target Padarn device and deploy all of the output files from your Solution.

To deploy your Padarn Solution:

1. Ensure your Padarn Evaluation Kit hardware is powered on and has an Ethernet connection on the same network as your development PC.
2. Determine the IP address of your target Padarn device. This is typically done by either double-clicking the network connection icon in the tray of the desktop or through the Network Connections Control Panel Applet.
3. Run **ConmanClient2.exe** on your target Padarn device. Padarn Evaluation Kit hardware will have a shortcut to this file on the desktop.
4. Run **CMAccept.exe** on your target Padarn device. Padarn Evaluation Kit hardware will have a shortcut to this file on the desktop.
5. From the Visual Studio menu select **Tools -> Options** to display the **Options** dialog.
6. Expand the **Device Tools** option node and select **Devices**.

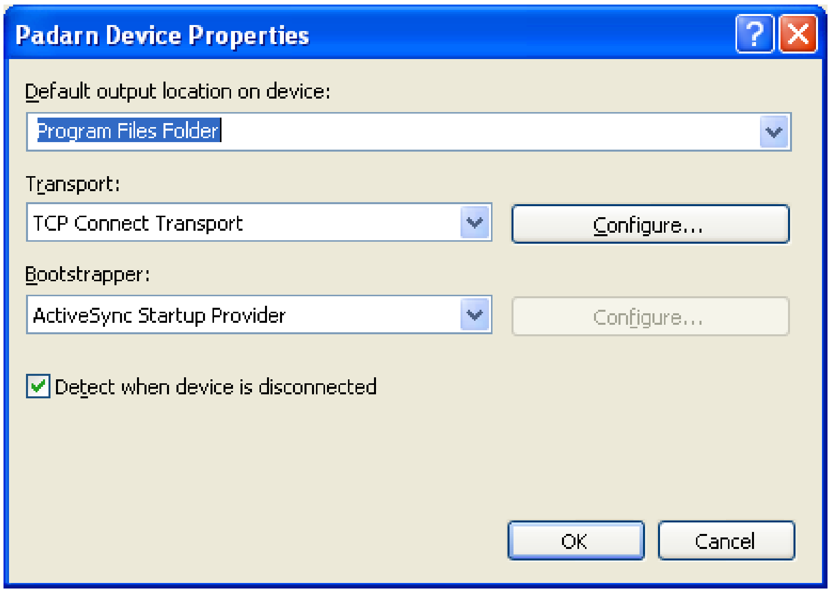


Show devices for platform: Windows CE

Devices: Windows CE Device

Default device: Windows CE Device

7. Select **Windows CE** from the **Show devices for platform** drop-down.
8. Select **Windows CE Device** from the **Devices** list.
9. Click on the **Save As...** Button.
10. Enter '**Padarn Device**' in the **Save As** Dialog and click **OK**.
11. Select the newly-created **Padarn Device** from the **Devices** list.
12. Click on the **Properties...** Button to display the **Padarn Device Properties** Dialog.



Padarn Device Properties

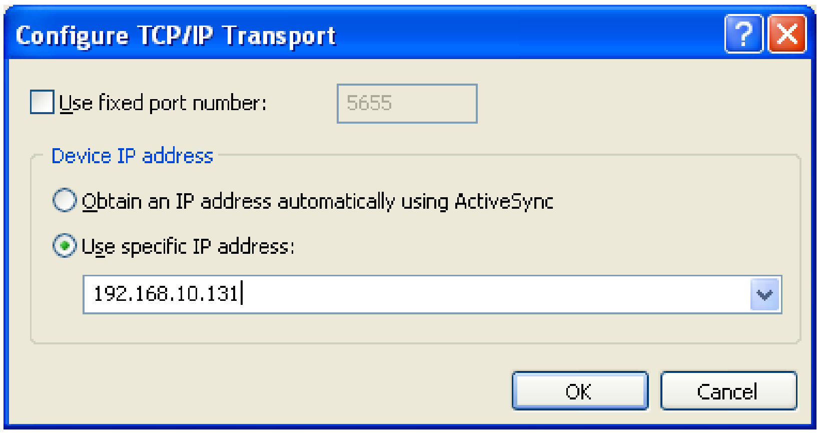
Default output location on device: Program Files Folder

Transport: TCP Connect Transport

Bootstrapper: ActiveSync Startup Provider

[✓] Detect when device is disconnected

13. Click on the **Configure** button for the TCP Connect Transport to display the Configure TCP/IP Transport Dialog.
14. Select the **Use Specific IP Address** option form the **Device IP address** options group.
15. Enter your device's IP address in the IP address textbox. Note that the IP address shown below is <u>only a representative number</u>. You need to enter the actual IP address of your specific device as determined in Step 2.



Configure TCP/IP Transport

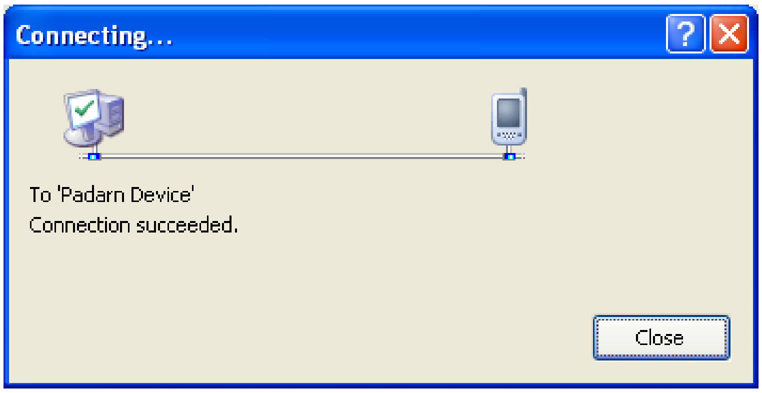
[ ] Use fixed port number: 5655

Device IP address

○ Obtain an IP address automatically using ActiveSync

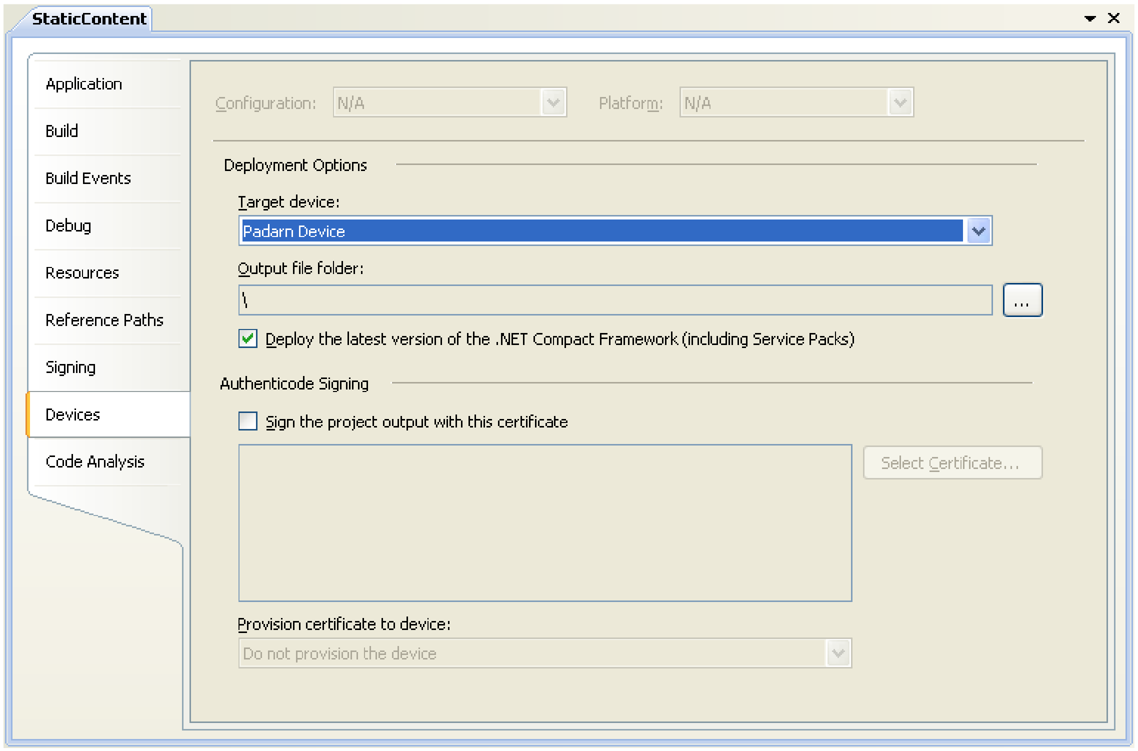
● Use specific IP address: 192.168.10.131

16. Click the **OK** button to three times to close all Dialogs and save your settings.
17. From the Visual Studio menu select **Tools -> Connect to Device** to display the **Connect To Device** dialog.
18. Select **Windows CE** from the **Platform** drop down.
19. Select **Padarn Device** from the **Devices** list.
20. Click the **Connect** Button to test your device connection.


Connecting...

To 'Padarn Device'
Connection succeeded.

21. Click **Close** to close the **Connecting…** Dialog.
22. In Visual Studio's **Solution Explorer** Pane, right-click on the **StaticContent** project and select **Properties** to display the **Static Content Properties** dialog.
23. Select the **Devices** Property Group.
24. Select Padarn Device from the Target Device drop-down.


StaticContent

Configuration: N/A    Platform: N/A

Deployment Options

Target device: Padarn Device

Output file folder: \

☑ Deploy the latest version of the .NET Compact Framework (including Service Packs)

Authenticode Signing

☐ Sign the project output with this certificate

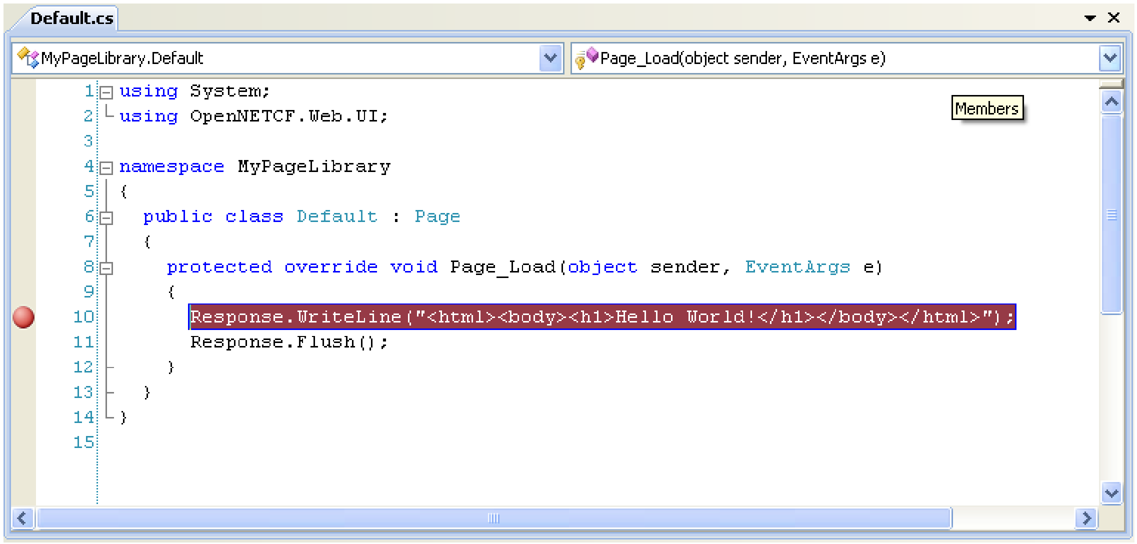
Provision certificate to device: Do not provision the device

25. **Repeat steps 22-24** for the **ocfhttpd** and **MyPageLibrary** projects.
26. In Visual Studio's **Solution Explorer** Pane, right-click on the **MyPadarnSolution** Solution and select **Deploy Solution** to deploy all of the Solution's output files to the target device.

## Exercise 5: Debugging the Page Code-Behind Assembly

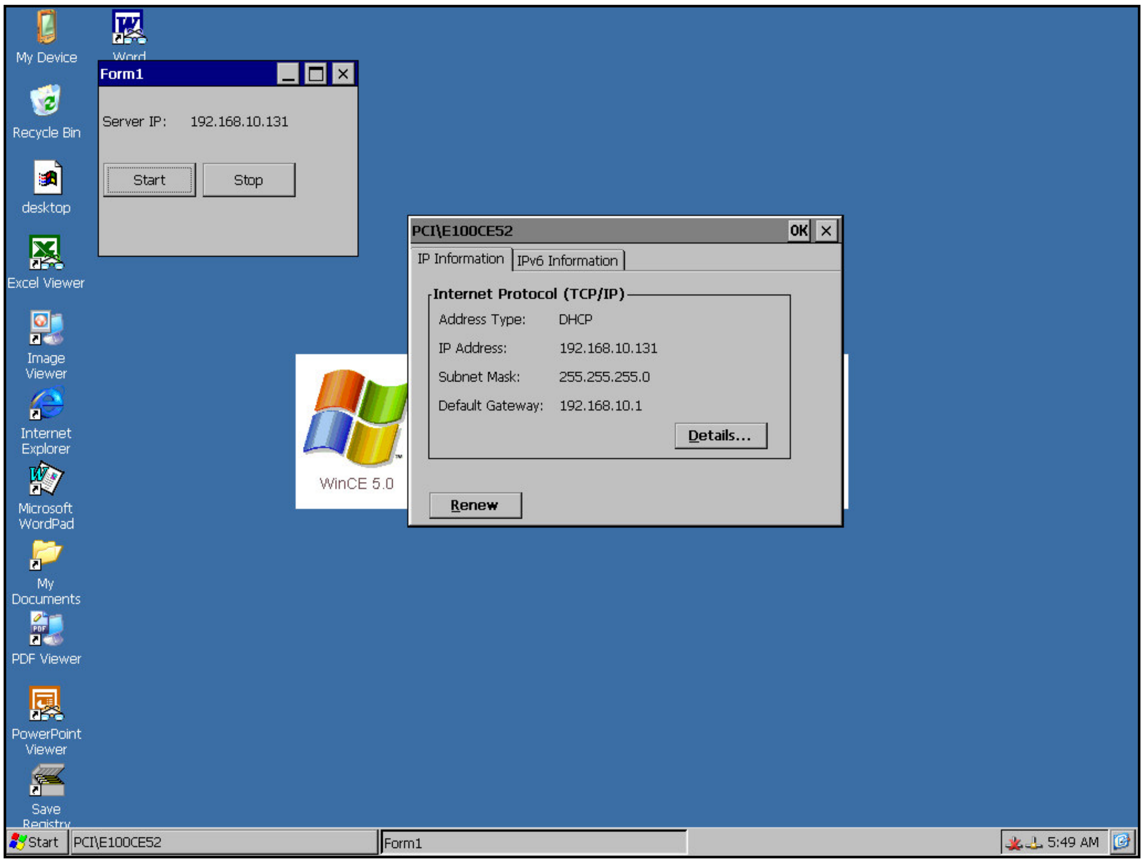
In this exercise you will set a breakpoint in your page's code-behind code, run the Padarn host application and see the debugger break at your break point.

1. In Visual Studio's **Solution Explorer** Pane, double-click on the **Default.cs** code file in the **MyPageLibrary** Project to open the **Default.cs** code file.
2. Place the cursor on line 10 (the line beginning with `Response.WriteLine`).
3. From the Visual Studio menu, select **Debug -> Toggle Breakpoint** to turn on a break point for this line.



```
 1  using System;
 2  using OpenNETCF.Web.UI;
 3
 4  namespace MyPageLibrary
 5  {
 6      public class Default : Page
 7      {
 8          protected override void Page_Load(object sender, EventArgs e)
 9          {
10              Response.WriteLine("<html><body><h1>Hello World!</h1></body></html>");
11              Response.Flush();
12          }
13      }
14  }
15
```

4. From the Visual Studio menu, select **Debug -> Start Debugging** to execute your Padarn hosting executable on the target device.



Form1

Server IP: 192.168.10.131

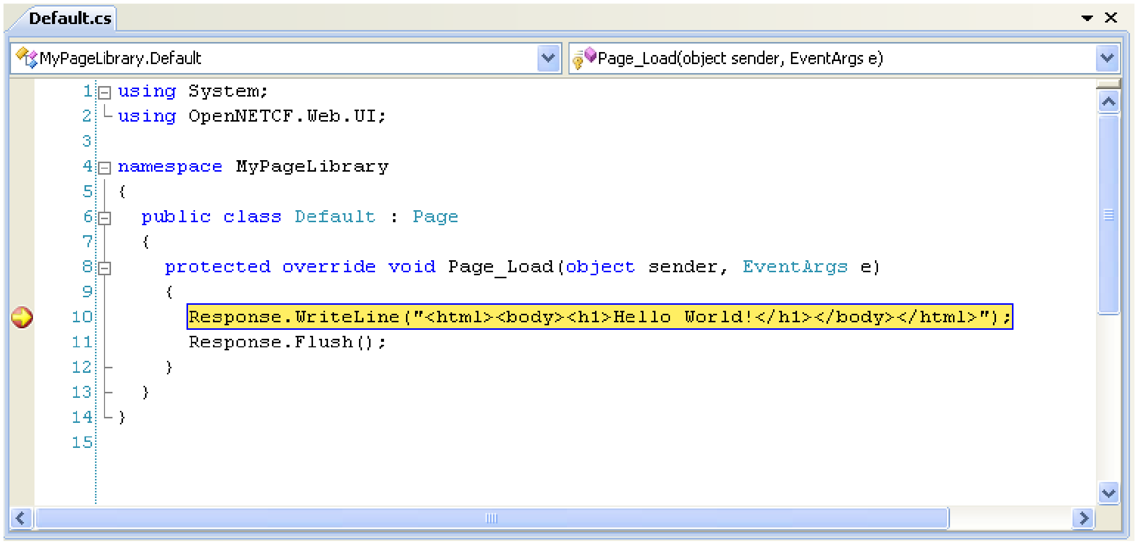
PCI\E100CE52

IP Information | IPv6 Information

Internet Protocol (TCP/IP)

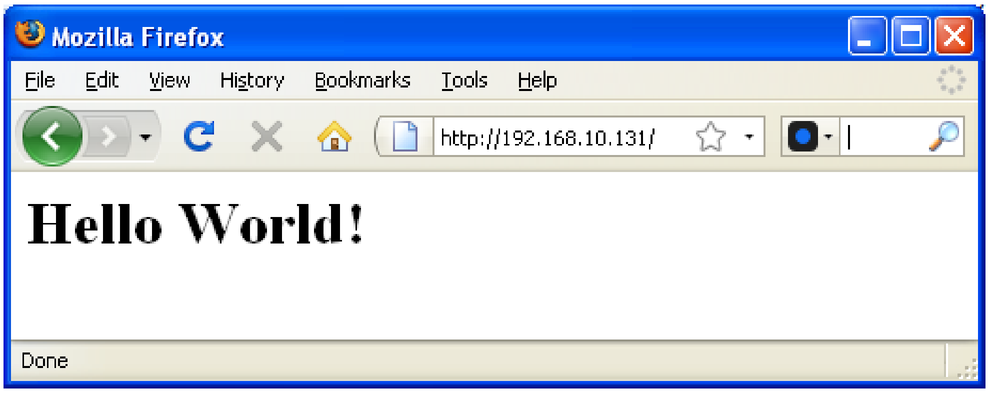
| | |
|---|---|
| Address Type: | DHCP |
| IP Address: | 192.168.10.131 |
| Subnet Mask: | 255.255.255.0 |
| Default Gateway: | 192.168.10.1 |

5. On the target Padarn Device, inside the **Form1** Form, click the **Start** Button to start the Padarn server.
6. On your PC launch your favorite browser application.
7. Navigate to the IP address of your target Padarn device.
8. Visual Studio will break execution at the breakpoint in **Default.cs** that you set in Step 3.

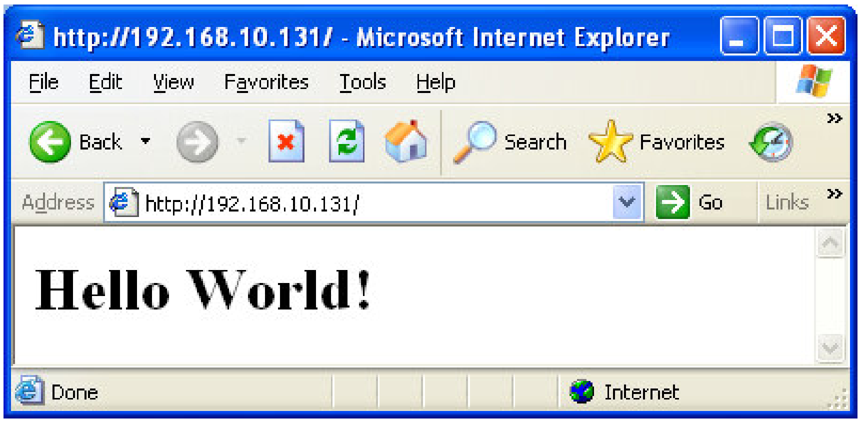


```
using System;
using OpenNETCF.Web.UI;

namespace MyPageLibrary
{
    public class Default : Page
    {
        protected override void Page_Load(object sender, EventArgs e)
        {
            Response.WriteLine("<html><body><h1>Hello World!</h1></body></html>");
            Response.Flush();
        }
    }
}
```

9. From the Visual Studio menu, select **Debug -> Continue** to continue execution of your Padarn code-behind assembly. Your browser will display your page contents.



Hello World!


http://192.168.10.131/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://192.168.10.131/ Go Links

Hello World!

## Hands-on Lab Summary

You now have seen all of the requisite steps for creating and debugging a Padarn web page.

In this lab you:

- ✓ Updated the Padarn Hosting Executable to create, start and stop the Padarn server
- ✓ Created a Padarn page ASPX file
- ✓ Created a code-behind class for your Padarn page
- ✓ Deployed your Padarn site to your target hardware
- ✓ Stopped the execution of your page's code-behind logic at a user break point

While the initial setup for the first page may seem fairly long, subsequent pages require much less setup (typically creation of just two files – the ASPX and the CS files).

The simple fact that you can set breakpoints in your page code and use all of Visual Studio's robust debugging tools greatly increases the speed at which you can develop and shortens the overall project development cycle when compared to developing ISAPI extensions or classic ASP 3.0 pages for the built-in server.

Also keep in mind that your page code can do anything that a managed application running on the device can do, including affecting hardware, using a local database or even calling remote web services.