

<http://projects.nikhilk.net>

Nikhil's Web Development Helper

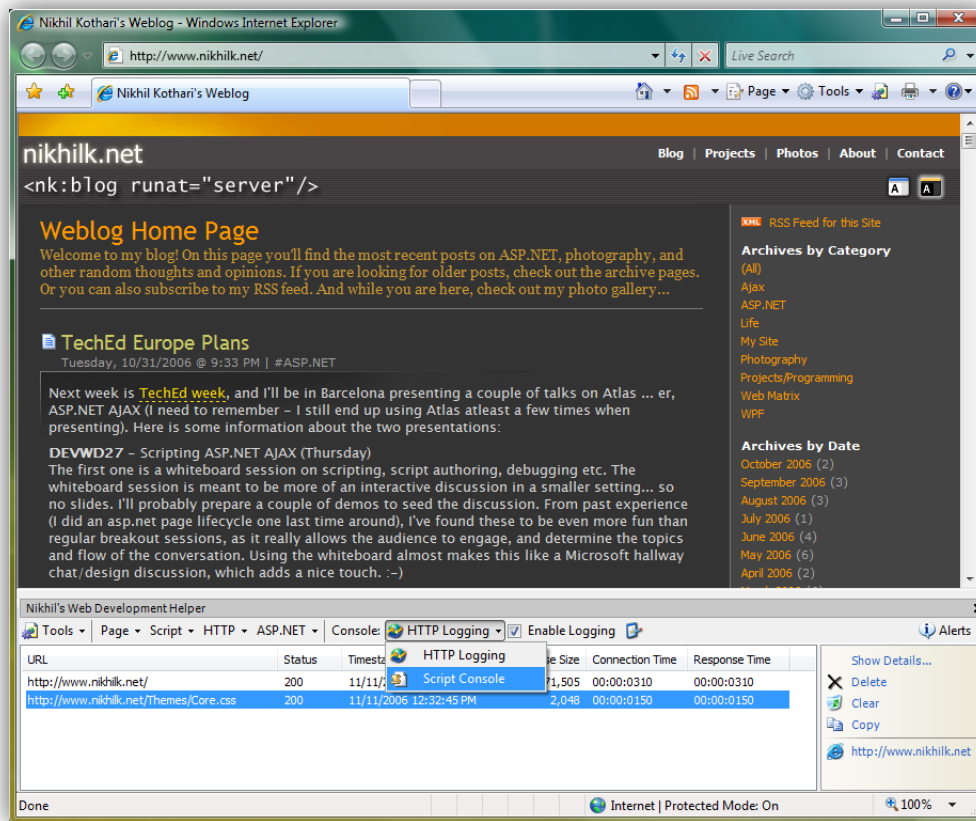
Version 0.8.5.0
July 24, 2007

Table of Contents

Introduction	3
Activating and Using Web Development Helper	3
HTTP Tracing	4
Scripting Tools.....	4
Page Tools	4
ASP.NET Tools	4
Using HTTP Tracing	4
Filtering HTTP Logging.....	6
Using the Scripting Tools.....	7
Error Reporting	7
Script Console	8
Script Immediate Window	8
debugService API for Script.....	8
Script Class Browser	10
Using the Page Tools.....	10
Using the ASP.NET Tools	11
View State Viewer	12
Trace Viewer	13
Cache Management	14
Restart Application	14
Credits	15
Version History.....	15
License.....	16

Introduction

Web Development Helper is a browser extension that plugs in to Internet Explorer to provide a useful set of utilities for the Web developer, esp. Ajax and ASP.NET developers. The utility provides functionality such as a DOM inspector, HTTP tracing and script diagnostics and immediate window.



The tool is currently available as a free download. Latest versions can be downloaded at <http://projects.nikhilk.net/Projects/WebDevHelper.aspx>. The same web site also provides additional information, and a form for submitting bug reports, feature requests and suggestions.

Activating and Using Web Development Helper

Web Development Helper has been tested on Internet Explorer 6.0 on Windows XP SP2, and Internet Explorer 7.0 on Windows XP SP2 and Windows Vista. Once the tool has been installed, and successfully registered with Internet Explorer, you will have the option to show a new Explorer Bar window that is docked to the bottom of the browser frame as shown in the above screenshot.

The Explorer Bar can be activated using the `Tools | Web Development Helper` command. If you choose to do so, you can also customize your browser's toolbar (right click on the toolbar, and choose `Customize`) to add a button for this command, especially if use it frequently (which is the general idea!)

The Explorer Bar associated with the tool has a command bar at the top which provides a menu and other commands that enable you to use the different features and functionality offered by the tool.

Commands are organized by feature area. The Explorer Bar currently contains two console windows – one for displaying HTTP trace logs, and another for displaying script trace messages and a script immediate window. The console can be switched using the Console dropdown on the command bar.

Finally, the tool provides a number of options. These can be customized via the Options command under the Tools menu that is present on the left-end of the command bar.

HTTP Tracing

The HTTP tracing feature allows logging HTTP(S) requests issued by the browser and scripts, and associated request/response details to allow analyzing network communication and usage.

Scripting Tools

The scripting tools provided by the extension enable improved script diagnostics and error tracking, as well as running and evaluating script snippets using a script console and immediate window.

Page Tools

The page tools allow inspecting the DOM, metadata, linked resources of the active page. The tool also allows capturing a screenshot of the entire page and saving it as a .png file.

ASP.NET Tools

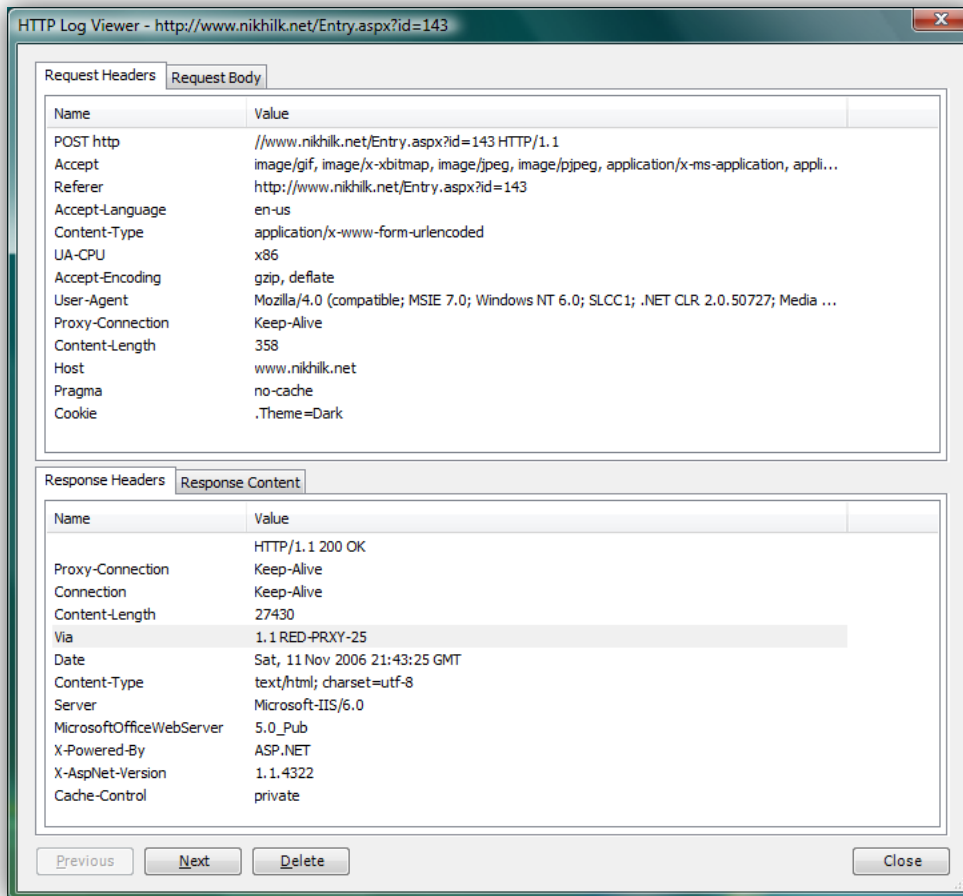
The ASP.NET tools work against local Web applications running in debug mode and provide the ability to visualize ASP.NET view state and trace content, manage cache usage and restart the ASP.NET app-domain.

Using HTTP Tracing

The HTTP tracing feature allows logging HTTP and HTTPS requests issued by the browser and scripts running within the browser. This allows analyzing network communication, viewing requests and the associated responses.

To start using this feature, you must first enable HTTP logging from the Web Development Helper command bar. It is a good idea to turn off logging when you don't need it. The tool does not require changing OS-level proxy settings. This has a couple of advantages. First it doesn't log HTTP requests made by other processes running on your machine. Secondly it is much easier to toggle it on and off as needed.

As requests are completed they are added as log entries in the HTTP Logging console showing the URL, response status code, time stamp, response size and connection times. You can click the Copy task for the selected entry to copy the URL of the entry to the clipboard. For any entry you can choose to see a more detailed view displaying all the request headers sent to the server, the request body (if the request was not a GET request), and the received response along with its headers.



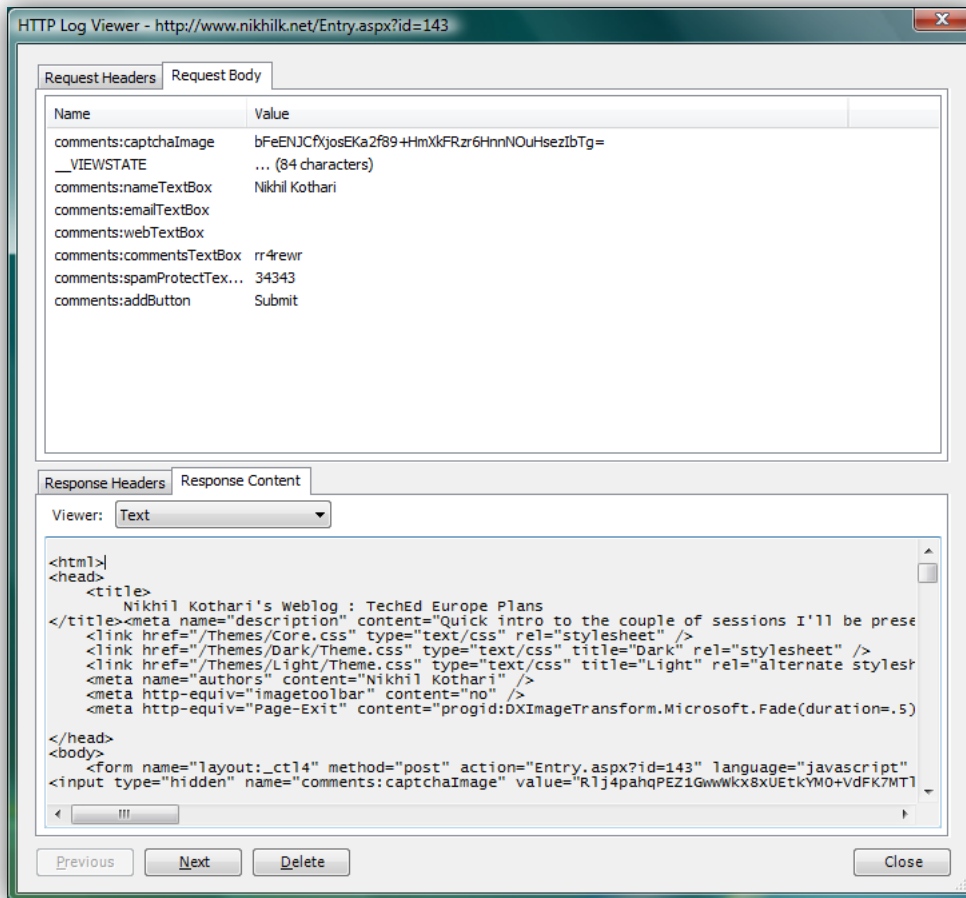
The above screenshot displays the request and response headers sent and received from the server respectively.

The following screenshot displays the request body and response sent and received from the server.

The tool tries to provide the best viewing experience. If the request body appears to be an HTML form post (like it is in the screenshot), the individual entries are parsed and displayed in name/value pair fashion. Otherwise the information is presented in a textbox as raw text¹.

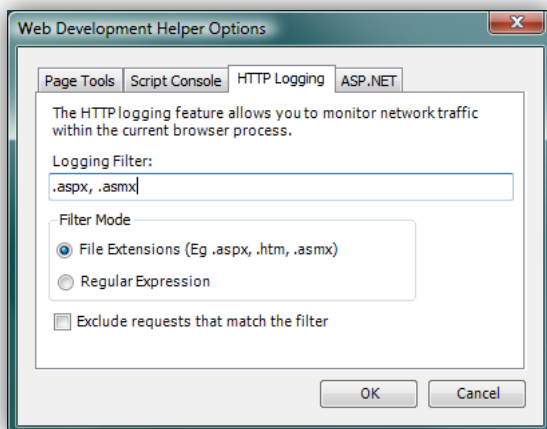
Similarly the tool provides the best view for the response. Textual responses are shown in text form (as shown in the screenshot). Binary responses are presented as a Hex view showing the individual bytes. Additionally the tool currently provides viewers for images (to display the image), JSON data (as a tree displaying the object graph), and a partial rendering response viewer (specific to partial rendering responses produced when using ASP.NET AJAX, and UpdatePanel controls).

¹ The plan is to add another viewer for the request that can display JSON data more effectively. This will be useful for AJAX applications such as those using ASP.NET AJAX using JSON serialization for data sent to the server.



Filtering HTTP Logging

By default all HTTP and HTTPS requests made by the browser and executing scripts within the same process are logged. You will likely want to choose a logging filter to avoid spurious entries resulting in noise in the logging window. You can access the filter settings via the Tools | Options command from the command bar.



For simple scenarios, requests can be filtered via file extension. For example, setting the logging filter to “.aspx, .asmx” displays .aspx and .asmx requests only. For more complex scenarios, you can provide a regular expression that is to match the URL of the request. In either case you can choose whether the filter you provide selects matching requests, or excludes matching requests for additional flexibility.

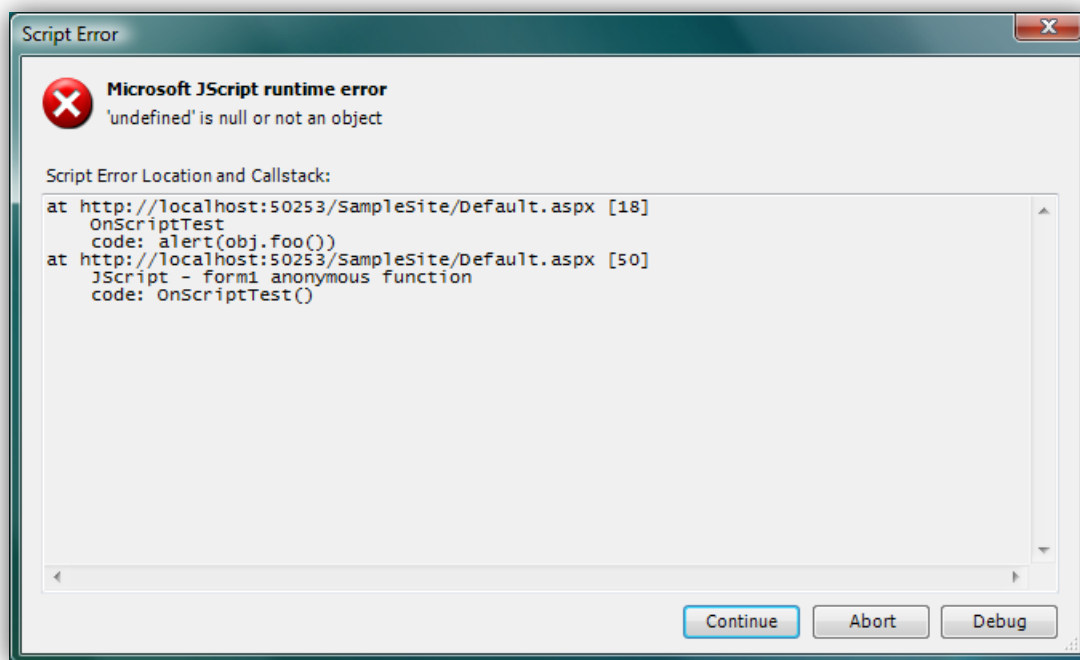
Using the Scripting Tools

The scripting tools provided by the extension enable improved script diagnostics and error messages, and script console and immediate window for capturing trace messages and evaluating script snippets against the active page. It also provides a script debug service for use by script within the page.

The script tools provide a quick-and-dirty debugging experience. This can be enabled by toggling on the Enable Debugging checkbox on the command bar. It should be noted that the tool does not provide a debugger in the true sense (Visual Studio provides an excellent debugging experience for Internet Explorer). However, the quick-and-dirty approach often suffices for a large class of scripting errors.

Error Reporting

The functionality provided by the tool includes better error reporting which translates more specifically into full call stack display, lines of code at each stack frame, correct line numbers, ability to resume execution or break into the debugger. The following screenshot displays the error dialog:



The Script Console tab in the Options dialog allows customizing whether this dialog is shown or errors are logged to the console without any modal user interface prompt.

Script Console

As alluded to, the tool also provides a script console. This console displays any debug trace being generated by the script. Internet Explorer provides the ability for script to output debug messages:

```
// JavaScript
Debug.writeln('Some debug text');
```

Various script frameworks provide cross-browser abstractions so you can use this capability without coding specifically to JavaScript in Internet Explorer.

```
// Microsoft ASP.NET AJAX2
debug.trace('Some debug text');

// Script#3 (as authored in c# before compilation to JavaScript)
Debug.WriteLine("Some debug text");
```

Script Immediate Window

Closely related to the script console window is the script immediate window. This window allows you to type in arbitrary script and execute it in the context of the current page. This allows you to run ad-hoc experiments without having to constantly change the page source and reload the page. You can also save and reload script snippets into this immediate window.

debugService API for Script

Finally, the tool exposes some new APIs for use by script. Web Development Helper adds a top-level “debugService” object to pages.

Traditionally outputting debug messages requires you to enable script debugging first. The debugService object provides a “trace” API that allows outputting debug messages to the script console, even if debugging is not enabled.

```
// JavaScript
if (window.debugService) {
    window.debugService.trace('Some debug message');
}
```

Note that the Debug object in Script# automatically does this.

Secondly script frameworks (like Microsoft ASP.NET AJAX and Script#) offer the ability to dump a JavaScript object in the form of trace messages to recursively drill into the object graph and show its properties and fields. The debugService API provides an “inspect” API that allows you to visualize this

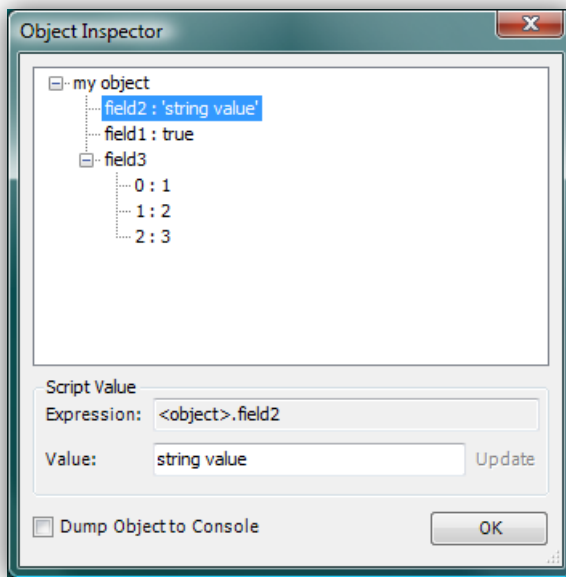
² <http://ajax.asp.net>

³ <http://projects.nikhilk.net/Projects/ScriptSharp.aspx> - Script# has been optimized to integrate the features of Web Development Helper transparently, so the calling code automatically benefits from the presence of the tool.

information in a tree view, and drill in on-demand, as opposed to generating a very large dump for a complex object graph. In addition the inspector window allows you to modify values.

```
// JavaScript
var obj = { field1: true, field2: "string value", field3: [ 1, 2, 3 ] };
if (window.debugService) {
    window.debugService.inspect('my object', obj);
}
```

Here is a screenshot of the inspector window:



In Script#, the framework automatically looks for debugService when your code calls Debug.Dump, and falls back to regular trace messages if the debugService is not available.

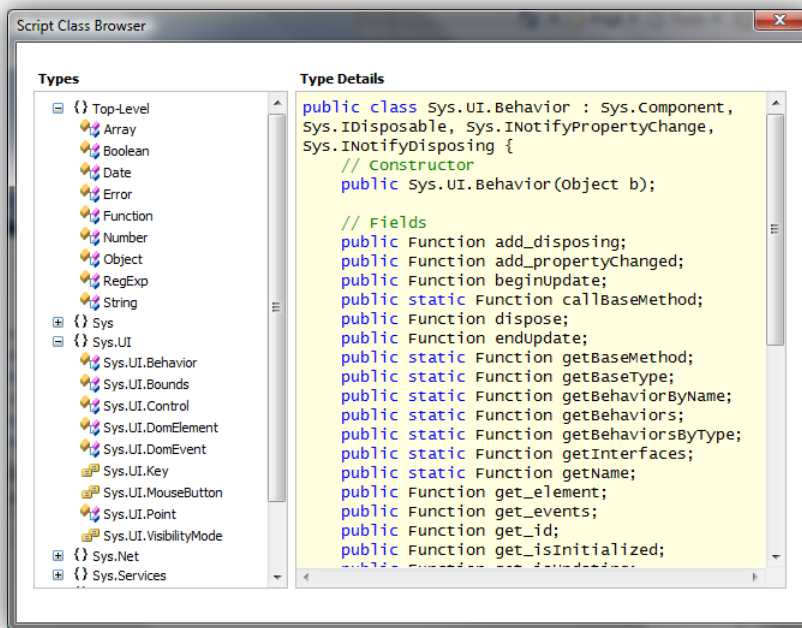
NOTE: In order to enable script debugging (for this tool, and Visual Studio) you need to make sure you have not disabled script debugging in Internet Explorer. Since that is the default setting, make sure you enable script debugging on your development machine. Please make sure the following three settings (within the Browsing category) available in the Advanced Settings tab of Internet Explorer's Tools | Options dialog are set appropriately as follows:

- Disable script debugging (Internet Explorer): Unchecked
- Disable script debugging (Other): Unchecked
- Display a notification about every script error: Checked
- [Optional] Show friendly HTTP error messages: Unchecked

Script Class Browser

The Script Class Browser allows browsing types (classes, interfaces and enumerations defined within namespaces) defined using the JavaScript type pattern implemented using Microsoft ASP.NET AJAX or Script#.

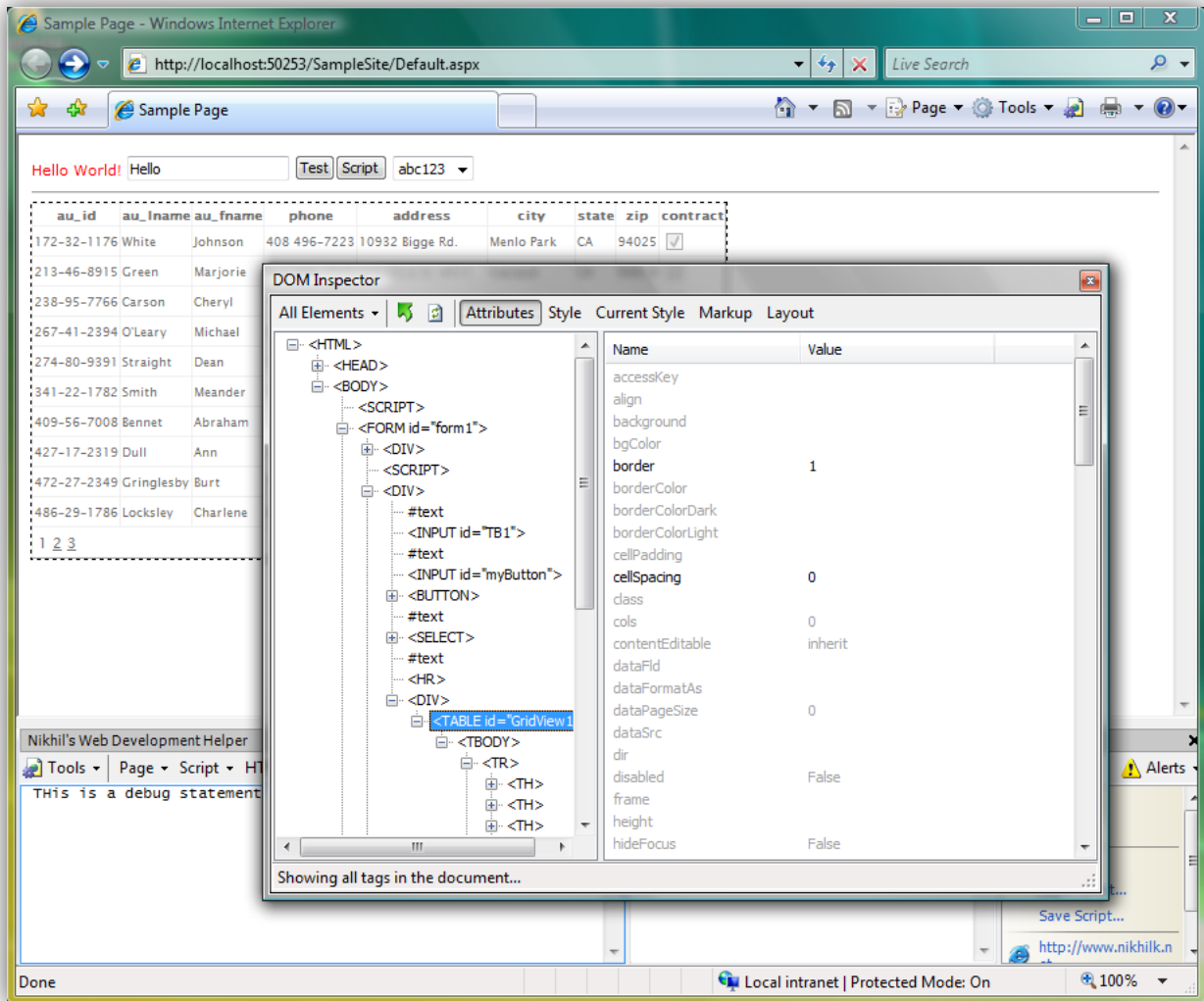
The class browser is still early in its implementation. It implements a simple tree of types distributed across namespaces, and a type definition is generated on the right pane, each time a type is selected within the tree. Below is a screen shot of the class browser.



Using the Page Tools

The page tools allow inspecting the DOM, metadata, linked resources of the active page. The tool also allows capturing a screenshot of the entire page and saving it as a .png file.

The DOM Inspector provides a view of the current state of the page as rendered by the browser. By default it allows drilling the hierarchy of HTML element nodes starting from the document element (<HTML>). Alternatively, if you have a selection in the document, the DOM Inspector can start the tree at the parent HTML element containing the selection. Finally, you can search for elements that match a tag name, an id or those associated with the specified CSS class to better scope the set of elements shown in the DOM inspector window. As shown in the screenshot, the DOM inspector highlights the selected element with a dashed border and allows viewing attribute values, style, markup etc. for the selected node



Using the ASP.NET Tools

The ASP.NET tools work against local Web applications running in debug mode and provide the ability to visualize ASP.NET view state and trace content, manage cache usage and restart the ASP.NET app-domain.

For security reasons the ASP.NET features only work with applications on `http://localhost`. This is usually a fine restriction, as you are likely to use the tool on your development machine. In order to enable the ASP.NET features you need to add an HTTP module to the default ASP.NET pipeline. This module is only active when your application is in debug mode, so it has minimal impact on applications. Here is an example `web.config` file that illustrates the entries you need to add (marked bold).

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <system.web>
```

```
<httpModules>
  <add name="DevInfo"
    type="nStuff.WebDevHelper.Server.DevInfoModule,
nStuff.WebDevHelper.Server, Version=0.5.0.0, Culture=neutral,
PublicKeyToken=8fc0e3af5abcb6c4" />
</httpModules>
<compilation debug="true"/>
</system.web>
</configuration>
```

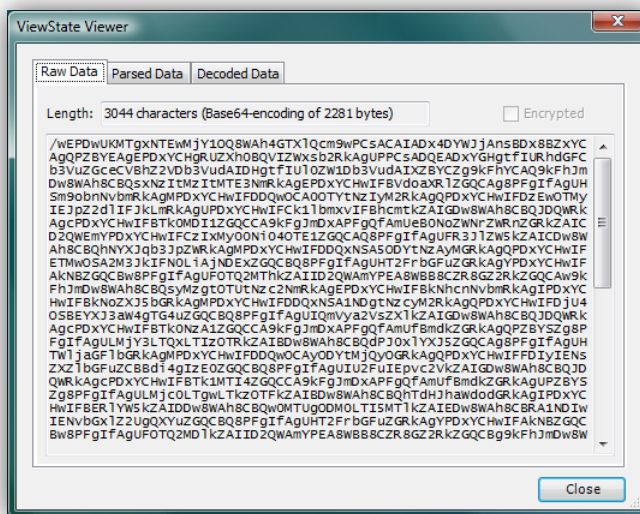
The nStuff.WebDevHelper.Server.dll module contains the HTTP module (installed into \Program Files\nStuff\Web Development Helper). You can either deploy this assembly into the bin directory of your Web application (if your application runs in full trust).

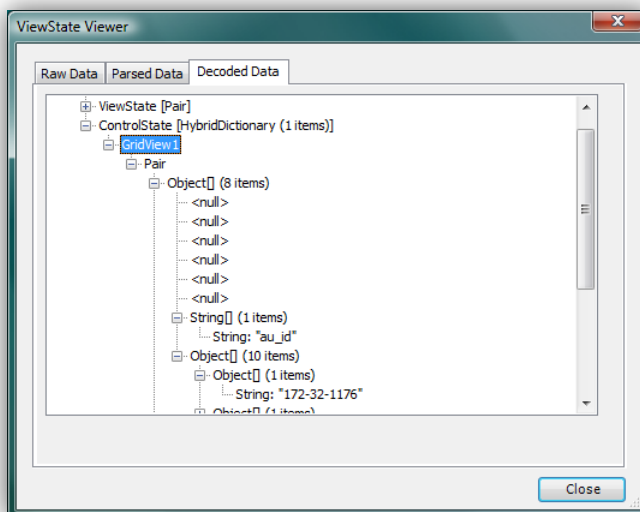
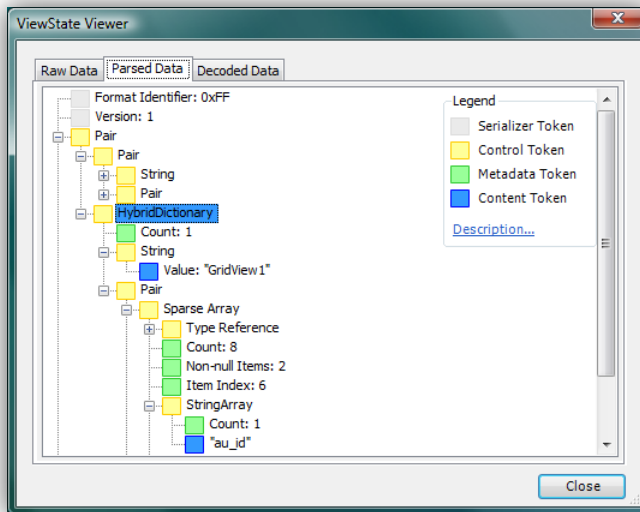
If your application is not configured to run in full trust, you must install this assembly to the GAC. You can open the \Windows\Assembly folder and drag in the nStuff.WebDevHelper.Server.dll.

Once you've enabled this module, a number of ASP.NET-specific features are enabled when you browse to a page from your Web application. These are available from the ASP.NET menu on the command bar.

View State Viewer

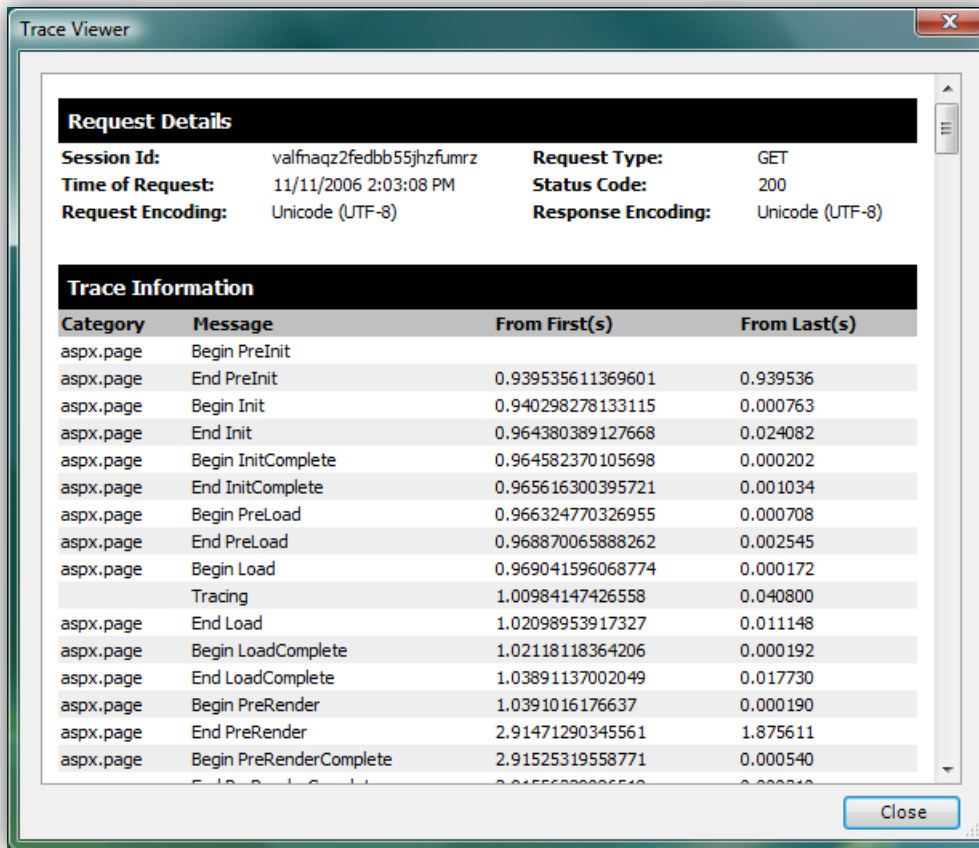
The view state viewer parses the viewstate hidden field to show three views: raw data, parsed data (interpretation of every byte in the view state field), and decoded data (or deserialized representation of actual data in view state), as shown by the following three screenshots.





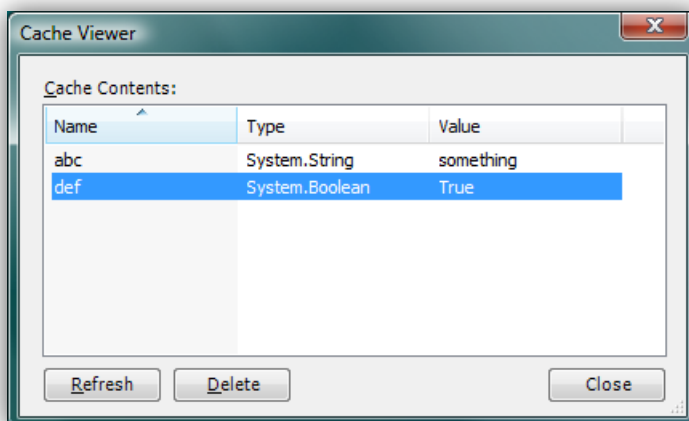
Trace Viewer

The tool hides trace information that is written into the page content by ASP.NET as it can interfere with layout. Instead it provides the same information in a separate dialog.



Cache Management

The tool allows you to view and manage the cache entries created by your application.



Restart Application

Often it is useful to be able to restart the application from a fresh state in a new app-domain. Rather than stopping the server, or making a config change to trigger this restart, the tool provides a command to simplify this.

Credits

I'd like to thank [Eilon Lipton](#), another developer on the ASP.NET team (and developer responsible for various ASP.NET AJAX features). Eilon, contributed the partial rendering response viewer used to display the data sent from ASP.NET in response to async postbacks, as well as provided the initial implementation of the Script Class Browser.

Version History

Version	Release Date	Notes
0.5	5/11/05	Initial version
0.5	5/12/05	Fixed bug where .css files were ignored by IE when using the Local Development Web Server. (Forgot to update version number)
0.6	5/17/05	Added Parsed ViewState view, Cache viewer, and ability to shutdown/restart application. NOTE: Assembly versions remain unchanged at 0.5 for now.
0.7	7/15/05	Added HTTP logging functionality
0.7.0.1	8/06/05	Bug Fixes UI works when opening IE windows (File New Window) HTTP request body display now works with XMLHTTP initiated requests.
0.7.1.0	8/19/05	New UI for HTTP logging, including display of response.
0.7.2.0	8/27/05	Console Window at the bottom of the browser and HTTP logging now uses that new window. Minor bug fixes.
0.7.5.0	8/30/05	Internal release (script console preview release)
0.8.0.0	9/19/05	DOM Inspector; (version demo'd at PDC05)
0.8.1.0	9/19/05	Fixed a bug where refreshing a page did not update the ASP.NET tools associated with the page.
0.8.1.1	9/20/05	Fixed various bugs: <ul style="list-style-type: none"> - Machines with older versions installed would show the new UI which is horizontally oriented in a vertical fashion - Show script call stack and source code in script error messages. - Show message about script debugging being disabled. - Other miscellaneous stuff.
0.8.2.0	11/15/05	The tool now runs against .NET 2.0 or VS 2005 RTM. New features: <ul style="list-style-type: none"> - Capture page to image (auto scrolls pages for a complete capture) - Page information to display metadata, links, referenced stylesheets etc. Improved DOM Inspector <ul style="list-style-type: none"> - Unobtrusive display of highlighted element - Walking up to parent element from DOM inspector)
0.8.3.0	6/26/06	The tool implements a debugService scriptable object which can be used by script running on the page to trace messages to the console window, or to inspect script objects in a window. This functionality is used by Script#. Some bugs fixed as well.
0.8.4.0	11/11/06	Enhancements to the HTTP tracing feature. (version demo'd at TechEd '06 Barcelona) <ul style="list-style-type: none"> - Added the ability to select a filter based on regular expressions (in addition to the previous file extensions-based) filter, as well as the ability to choose whether the filter is used to exclude matching requests or include requests. - Enhanced response viewers for image files, JSON responses, and ASP.NET AJAX partial rendering responses.

0.8.5.0	7/24/07	Added Script Class Browser feature Fixed installation issues on Vista. Fixed bugs in JSON view of the HTTP tracing functionality.
---------	---------	---

License

End User License Agreement for Web Development Helper

IT IS IMPORTANT THAT YOU CAREFULLY READ THIS NOTICE BEFORE INSTALLING THIS PRODUCT. BY INSTALLING, OR OTHERWISE USING THIS SOFTWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE AGREEMENT (THE "AGREEMENT") WHICH CONSTITUTES A LEGALLY BINDING CONTRACT BETWEEN THE LICENSOR (PROJECTS.NIKHILK.NET, HEREAFTER 'WE', OR 'US') AND THE LICENSEE (EITHER AN INDIVIDUAL OR ENTITY, HEREAFTER 'YOU').

THIS AGREEMENT

1.1 In this Agreement, the phrase "Software" means any version of the computer programs above and all associated media, printed materials, "online" or electronic documentation and bundled software.

1.2 The Software is licensed, not sold, to You for use only under the terms of this Agreement. We reserve any rights not expressly granted to You.

1.3 By installing, copying or otherwise using the Software, You agree to be bound by the terms of this Agreement. If You do not agree to the terms of this Agreement You must not use the Software and must immediately delete any and all copies of the Software in your possession.

GRANT OF LICENSE

2.1 We hereby grant You the following non-exclusive license to use the Software. The rights granted to the Licensee are personal and non-transferable.

2.2 The following are the restrictions placed on the use of the Software. You may not:

- Modify or adapt the Software into another program or product.
- Reverse engineer, disassemble or decompile, or make any attempt to discover the source code of the Software through current or future available technologies.
- Redistribute, publish or deploy the Software on a standalone basis for others to copy without prior acknowledgment from the Licensor.
- Copy or republish any portion of the documentation without prior acknowledgment from the Licensor.
- Sell, re-license, sub-license, rent, lease any part of the Software or create derivative works.
- Use the Software to perform any unauthorized transfer of information or any illegal purpose.

2.3 We may from time to time create updated versions of the Software and may, at our option, make such updates available to You.

2.4 The Software is pre-release software. We have the sole right to determine all aspects of

future updates, changes, and releases of the Software.

2.5 You permit the Software to connect and communicate with our servers to send version and usage information for the purposes of improving the Software or sending information about available updates.

2.6 You agree to indemnify, hold harmless, and defend Us from and against any claims, allegations, lawsuits, losses and costs (including attorney fees), that arise or result from the use, deployment or distribution the software.

2.7 Any feedback including bug reports, feature suggestions or ideas provided by You to Us through any communication channel are given to Us without any associated charge or implied patent or intellectual rights. Thereafter, We have the full right to use, share and commercialize such feedback in any way and for any purpose. You will not give feedback that is subject to a license that requires Us to license the Software to third parties because of inclusion of such feedback. These rights survive this Agreement.

2.8 We do not provide any support services because the software is being made available to You in "as-is" form.

2.9 We reserve the right to update the Agreement and the terms of the License with newer versions of the Software.

INTELLECTUAL PROPERTY RIGHTS

3.1 The Software is protected by copyright and other intellectual property laws. Title to, ownership of, and all rights and interests in each and every part of the Software (including all copyrights, trademarks, patent rights or other intellectual property rights of whatever nature), and all copies thereof shall remain at all times vested in Us.

WARRANTIES

4.1 We expressly disclaim any warranty for the Software. The Software and any associated materials are provided "As Is" without warranty of any kind, either express or implied, including without limitation, the implied warranties or merchantability, fitness for a particular purpose, or non-infringement. The entire risk arising out of use or performance of the Software remains with You.

TERMINATION

5.1 This Agreement takes effect upon your use of the Software and remains effective until terminated. You may terminate it at any time by destroying all copies of the Software in possession. It will also automatically terminate if You fail to comply with any term or condition of this Agreement. You agree on termination of this Agreement to destroy all copies of the Software in possession.

GENERAL TERMS

6.1 This written Agreement is the exclusive agreement between You and Us concerning the Software and supersedes any prior agreement, communication, advertising or representation concerning the Software.

6.2 This Agreement may be modified only by a writing signed by You and Us.

6.3 In the event of litigation between You and Us concerning the Software, the prevailing party in the litigation will be entitled to recover attorney fees and expenses from the other party.

6.4 This Agreement is governed by the laws of the State of Washington, USA. Irrespective of the country in which the Software was acquired, the construction, validity and performance of the Agreement shall be governed in all respects by English law. You agree to submit to exclusive jurisdiction of English courts.

6.5 If any provision of this Agreement is found to be invalid by any court having competent jurisdiction, the invalidity of such provision shall not affect the validity of the remaining provisions of this Agreement, which shall remain in full force and effect.

6.6 You agree that the Software will not be shipped, transferred or exported into any country or used in any manner prohibited by the United States Export Administration Act or any other export laws, restrictions or regulations.