Debugging with Fiddler

The complete reference from the creator of the Fiddler Web Debugger

This is a SAMPLE containing the Table of Contents and a bit of content so you can decide whether the book meets your needs and renders nicely on your device.

Buy the book in paperback or ebook format at http://www.fiddlerbook.com

Eric Lawrence
Debugging with Fiddler
Cover Photo: Nicholas Wave; ©IStockPhoto.com/@by_nicholas
Everything else: ©2012 Eric Lawrence. All rights reserved. Please don’t pirate this book in whole or in part. Beyond the nine years I’ve spent developing Fiddler, I spent nine months on this book and I’d like to be able to pay for the coffee I drank while writing it. :)

Sample Version LULU 1.00 / Fiddler Version 2.3.9.9

Legalese
Trademarks mentioned in this book are (obviously) the property of their respective owners, and are only used to identify the products or services mentioned.

This book is provided “as is.” In no event shall I, the author, be liable for any consequential, special, incidental or indirect damages of any kind arising out of the delivery, accuracy, or use of this book. This book was written with care, but no one warrants that it is error-free. On the contrary, I guarantee that this book contains at least a few errors, and I promise to be suitably embarrassed when you point them out to me (http://fiddlerbook.com/errata) so that I may update the next version.

e_lawrence@hotmail.com
@ericlaw on Twitter
# Table of Contents

Acknowledgements ............................................................................................................... iii

Table of Contents .................................................................................................................. iii

Introduction ............................................................................................................................ 1

Origins ....................................................................................................................................

About this book ....................................................................................................................... 2

A Quick Primer ....................................................................................................................... 5

Basic Concepts ...................................................................................................................... 9

Usage Scenarios .................................................................................................................... 12

An Incomplete List of Things Fiddler Can Do ..................................................................... 23

An Incomplete List of Things Fiddler Cannot Do ................................................................. 28

Exploring Fiddler .................................................................................................................. 32

Getting Started .....................................................................................................................

System Requirements .......................................................................................................... 35

Installing Fiddler .................................................................................................................. 40

Permissions and XCOPY Deployment ................................................................................... 44

Updating Fiddler .................................................................................................................. 48

Uninstalling Fiddler ............................................................................................................... 52

The Fiddler User-Interface .................................................................................................... 55

The Web Sessions List ......................................................................................................... 59

Understanding Icons and Colors ...................................................................................... 63

Keyboard Reference ............................................................................................................. 66

Web Sessions Context Menu ............................................................................................... 69

Fiddler’s Main Menu ............................................................................................................ 73

The File Menu ....................................................................................................................... 77

The Edit Menu ....................................................................................................................... 81

The Rules Menu .................................................................................................................... 84

Performance Submenu ........................................................................................................ 87

The Tools Menu .................................................................................................................... 90

The View Menu .................................................................................................................... 93

The Help Menu .................................................................................................................... 96

Fiddler’s About Box ............................................................................................................. 99

Fiddler’s Toolbar ................................................................................................................... 103

3 | Introduction
Fiddler's Status Bar

QuickExec

QuickExec Selection Commands
Default FiddlerScript Commands

Application Hotkeys

Statistics Tab

The Filters tab

Hosts
Client Process
Request Headers
Breakpoints
Response Status Code
Response Type and Size
Response Headers

The Timeline tab

Mode: Timeline
Mode: Client Pipe Map
Mode: Server Pipe Map
Using the Timeline for Performance Analysis

The AutoResponder tab

Specifying the Match Condition
Matching Against Request Bodies
Specifying the Action Text
Using RegEx Replacements in Action Text
Drag-and-Drop support
FARX Files

The TextWizard

Character Encodings

The Composer tab

Request Options
Raw Requests
Parsed Requests
Issuing Sequential Requests
File Upload Requests

Automatic Request Breakpoints

The Log tab

The Find Sessions Window

The Host Remapping Tool

TECHNIQUES AND CONCEPTS

Retargeting Traffic with Fiddler

Method #1 - Rewriting
Method #2 - Rerouting
Method #3 - Redirecting

Features to Retarget Requests

Comparing Sessions

UltraDiff
Comparing Multiple Sessions at Once

Debugging with Breakpoints

Setting Breakpoints
Tampering Using Inspectors
The Breakpoint Bar
Resuming Multiple Sessions

CONFIGURING FIDDLER AND CLIENTS

Fiddler Options

General Options
HTTPS Options
Extensions Options
Connections Options
Appearance Options

HeaderEncoding Setting
Preferences
Configuring Clients
Capturing Traffic from Browsers
Firefox
Opera
Other Browsers
Capturing Traffic from Other Applications

WinHTTP

.NET Framework

Java

PHP / CURL

Capturing Traffic from Services

Capturing Traffic to Loopback

Loopback Bypasses

Loopback Authentication

Loopback Blocked from Metro-style Windows 8 Apps

Running Fiddler on Mac OSX

Capturing Traffic from Other Computers

Capturing Traffic from Devices

Apple iOS Proxy Settings

Windows Phone Proxy Settings

Windows RT Proxy Settings

Other Devices

Using Fiddler as a Reverse Proxy

Acting as a Reverse Proxy for HTTPS

Chaining to Upstream Proxy Servers

Chaining to SOCKS / TOR

VPNs, Modems, and Tethering

DirectAccess

Memory Usage and Fiddler’s Bitness

Buffering vs. Streaming Traffic

Request Buffering

Response Buffering

COMET

HTML5 WebSockets

Fiddler and HTTPS

Trusting the Fiddler Root Certificate

Machine-wide Trust on Windows 8

Manually Trusting the Fiddler Root
Raw .......................................................................................................................... 
SyntaxView .................................................................................................................. 
TextView ....................................................................................................................... 
Transformer ................................................................................................................... 
  Background on Response Encodings .................................................................. 
  Adding and Removing Encodings using the Transformer ................................. 
  Other Ways to Remove Encodings ................................................................. 
WebForms .................................................................................................................... 
WebView ..................................................................................................................... 
XML ............................................................................................................................. 
EXTENSIONS ............................................................................................................. 
Overview ..................................................................................................................... 
  Popular 3rd Party Extensions ........................................................................... 
    Performance Add-ons ............................................................................... 
    Security Add-ons ....................................................................................... 
  Extensions I’ve Built ......................................................................................... 
JavaScript Formatter ................................................................................................. 
Gallery ....................................................................................................................... 
  Full-Screen View ............................................................................................... 
Content Blocker ......................................................................................................... 
Traffic Differ ............................................................................................................... 
FiddlerScript Editors ................................................................................................. 
  FiddlerScript Tab ............................................................................................... 
  ClassView Sidebar ............................................................................................ 
  Fiddler2 ScriptEditor ........................................................................................ 
SAZClipboard ............................................................................................................ 
AnyWHERE ................................................................................................................ 
STORING, IMPORTING, AND EXPORTING TRAFFIC .......................................... 
Session Archive Zip (SAZ) Files ............................................................................. 
  Protecting SAZ Files ......................................................................................... 
FiddlerCap ................................................................................................................ 
  Capture Box ....................................................................................................... 
  Capture Options Box .........................................................................................
# FiddlerScript and Automation Tools

## FiddlerScript Functions

- Extending Fiddler with FiddlerScript
  - About FiddlerScript
  - Editing FiddlerScript
    - Updating FiddlerScript at Runtime
    - Resetting to the Default FiddlerScript
  - FiddlerScript Functions
    - Session Handling Functions
      - OnPeekAtRequestHeaders
      - OnBeforeRequest
      - OnPeekAtResponseHeaders
      - OnBeforeResponse
      - OnReturningError
    - General Functions
      - Main
      - OnRetire
      - OnBoot
      - OnShutdown
      - OnAttach
      - OnDetach
      - OnExecAction(sParams: string[])

## Importing and Exporting Sessions

- Import Formats
- Export Formats
  - HTML5 AppCache Manifest
  - HTTPArchive v1.1 and v1.2
  - MeddlerScript
  - Raw Files
  - Visual Studio WebTest
  - WCAT Script
Quiet Mode
Driving Fiddler from Batch Scripts
Driving Fiddler from Native or .NET Code
Extending Fiddler’s UI - Menus
Extending the Tools Menu
Extending the Web Sessions Context Menu
Extending the Rules Menu
  Boolean-bound Rules
  String-bound Rules
Creating New Top-Level Menus,
Extending Fiddler’s UI - Adding Columns to the Web Sessions List
Binding Columns using Attributes
Binding Columns using AddBoundColumn
FiddlerObject Functions
  FiddlerObject.ReloadScript()
  FiddlerObject.StatusText
  FiddlerObject.log(sTextToLog)
  FiddlerObject.playSound(sSoundFilename)
  FiddlerObject.flashWindow()
  FiddlerObject.alert(sMessage)
  FiddlerObject.prompt(sMessage)
  FiddlerObject.createDictionary()
  FiddlerObject.WatchPreference(sPrefBranch, oFunc)
Referencing Assemblies
Example Scripts
Request Scripts
  Add (or Overwrite) a Request Header
  Remove Request Headers
  Flag Requests that Send Cookies
  Rewrite a Request from HTTP to HTTPS
  Swap the Host Header
  Drop a Connection
  Prevent Response Streaming
EXTENDING FIDDLER WITH .NET CODE

Extending Fiddler with .NET

Project Requirements and Settings

Debugging Extensions

Best Practices for Extensions

Best Practice: Use an Enable Switch
Best Practice: Use Delay Load
Best Practice: Beware “Big Data”
Best Practice: Use the Reporter Pattern for Extensions

Interacting with Fiddler’s Objects

The Web Sessions List

Session[] GetAllSessions()
Session GetFirstSelectedSession()
Session[] GetSelectedSessions()
Session[] GetSelectedSessions(int iMax)
void actSelectAll()
void actSelectSessionsMatchingCriteria(doesSessionMatchCriteriaDelegate oDel)
void actRemoveSelectedSessions()
void actRemoveUnselectedSessions()
bool actLoadSessionArchive(string sFilename)
void actSaveSessionsToZip()
void actSaveSessionsToZip(string sFilename, string sPwd)
void actSessionCopyURL()
void actSessionCopySummary()
void actSessionCopyHeadlines()
int FiddlerApplication.UI.lvSessions.SelectedItems
SimpleEventHandler FiddlerApplication.UI.lvSessions.OnSessionsAdded

Response Scripts

Hide Sessions that Returned Images
Flag Redirections
Replace Text in Script, CSS, and HTML
Remove All DIV Elements

More Examples
Session Objects

Sending Strings to the TextWizard

Session Objects

oRequest

requestBodyBytes

oResponse

responseBodyBytes

oFlags

void Abort()

bBufferResponse

bHasResponse

bypassGateway

clientIP

clientPort

bool COMETPeek()

fullUrl

PathAndQuery

port

host

hostname

bool Hostnamels(string)

bool HTTPMethodIs(string)

bool uriContains(string)

id

isFTP

isHTTPS

isTunnel

LocalProcessID

bool utilDecodeRequest()

bool utilDecodeResponse()

responseCode

state

BitFlags

Timers

Sending Strings to the TextWizard
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming with Preferences</td>
<td></td>
</tr>
<tr>
<td>Preference Naming</td>
<td></td>
</tr>
<tr>
<td>The IFiddlerPreferences Interface</td>
<td></td>
</tr>
<tr>
<td>Storing and Removing Preferences</td>
<td></td>
</tr>
<tr>
<td>Internally, all preference values are stored as strings; the</td>
<td></td>
</tr>
<tr>
<td>Retrieving Preferences</td>
<td></td>
</tr>
<tr>
<td>Watching for Preference Changes</td>
<td></td>
</tr>
<tr>
<td>Notifications in Extensions</td>
<td></td>
</tr>
<tr>
<td>Notifications in FiddlerScript</td>
<td></td>
</tr>
<tr>
<td>Building Extension Installers</td>
<td>7</td>
</tr>
<tr>
<td>Building Inspectors</td>
<td></td>
</tr>
<tr>
<td>Inspecting the Session Object</td>
<td>8</td>
</tr>
<tr>
<td>Dealing with HTTP Compression and Chunking</td>
<td></td>
</tr>
<tr>
<td>Decoding a Copy of the Body</td>
<td></td>
</tr>
<tr>
<td>Using the GetRe*BodyAsString Methods</td>
<td></td>
</tr>
<tr>
<td>Using the utilDecode* Methods</td>
<td></td>
</tr>
<tr>
<td>Inspector Assemblies</td>
<td></td>
</tr>
<tr>
<td>Building Extensions</td>
<td></td>
</tr>
<tr>
<td>Understanding Threading</td>
<td></td>
</tr>
<tr>
<td>Integrating with QuickExec</td>
<td></td>
</tr>
<tr>
<td>Example Extension</td>
<td></td>
</tr>
<tr>
<td>Extension Assemblies</td>
<td></td>
</tr>
<tr>
<td>Building Import and Export Transcoders</td>
<td></td>
</tr>
<tr>
<td>Direct Fiddler to load your Transcoder assemblies</td>
<td></td>
</tr>
<tr>
<td>The ProfferFormat Attribute</td>
<td></td>
</tr>
<tr>
<td>The ISessionImporter Interface</td>
<td></td>
</tr>
<tr>
<td>The ISessionExporter Interface</td>
<td></td>
</tr>
<tr>
<td>Handling Options</td>
<td></td>
</tr>
<tr>
<td>Providing Progress Notifications</td>
<td></td>
</tr>
<tr>
<td>Notes on Threading and Transcoders in FiddlerCore</td>
<td></td>
</tr>
<tr>
<td>Beyond Files</td>
<td></td>
</tr>
</tbody>
</table>
Example Transcoder

FiddlerCoreApplication Properties and Fields

FiddlerCoreApplication Events

FiddlerCoreApplication StartupFlags

Compiling the Sample Application

The FiddlerCoreApplication Class

FiddlerCoreApplication Methods

Legalities

Getting Started with FiddlerCoreApplication

Overview

Introduction

14 | Introduction
isClosing.................................................................................................................................
Log...........................................................................................................................................

oDefaultClientCertificate ........................................................................................................
oProxy ........................................................................................................................................

oTranscoders ...........................................................................................................................
Prefs ...........................................................................................................................................

The Rest of the Fiddler API ......................................................................................................

Common Tasks with FiddlerCore ...........................................................................................

Keeping track of Sessions ........................................................................................................

Getting Traffic to FiddlerCore .................................................................................................

Trusting the FiddlerCore Certificate ....................................................................................... 

Generating Responses ............................................................................................................

Other Resources ......................................................................................................................

APPENDICES ........................................................................................................................

Appendix A: Troubleshooting .................................................................................................

Missing Traffic ...........................................................................................................................

Interference from Security Software ....................................................................................... 

Problems Downloading Fiddler ..............................................................................................

Problems Installing Fiddler ......................................................................................................

Problems Running Fiddler ........................................................................................................

Corrupted Proxy Settings ........................................................................................................

Resetting Fiddler .....................................................................................................................

Troubleshooting Certificate Problems ...................................................................................

Wiping all traces of Fiddler ....................................................................................................... 

Fiddler crashes complaining about the "Configuration System" ..............................................

Fiddler randomly stops capturing traffic ................................................................................

Fiddler stalls when streaming RPC-over-HTTPS traffic .........................................................

Appendix B: Command Line Syntax ....................................................................................... 

Option Flags .............................................................................................................................

Examples ...................................................................................................................................

Appendix C: Session Flags ......................................................................................................

Session Display Flags .............................................................................................................

Breakpoint and Editing Flags .................................................................................................
Networking Flags .................................................................
Authentication Flags ..........................................................
Client Information Flags ....................................................
Performance Simulation Flags ............................................
HTTPS Flags ..................................................................
Request Composer Flags ...................................................
Other Flags ......................................................................

Appendix D: Preferences ....................................................
Network Preferences ........................................................
HTTPS Preferences ..........................................................
Fiddler UI Preferences ......................................................
FiddlerScript Preferences ..................................................
TextWizard Preferences ....................................................
Request Composer Preferences .........................................
Path Configuration ...........................................................
Miscellaneous ................................................................
Extension Preferences ......................................................
Raw Inspector .................................................................
JavaScript Formatter ........................................................
Certificate Maker ............................................................

Index .............................................................................
Introduction
About this book

After nearly 9 years and one hundred version updates, Fiddler has evolved into a powerful utility and platform that can perform a wide variety of tasks. It has a rich extensibility model and a community of add-on developers who have broadened its usefulness as a performance, security, and load-testing tool. Questions in email, online discussion groups, and numerous conferences over the years made it overwhelmingly apparent that most users only exploit a tiny fraction of Fiddler’s power. I came to realize that thousands of users would get a lot more out of Fiddler if there were a complete reference to the tool available. This book is the product of that realization.

As Fiddler’s developer, I’ve found it both easy and challenging to write this book. It’s easy, because I understand Fiddler deeply, down to its very foundation, and can consult the source code to research obscure details. On the other hand, it’s been very challenging, as every time I choose an interesting scenario or feature to write about, I’m forced to think deeply about that scenario or feature. Commonly, I’ve found myself developing improvements to revise Fiddler and minimize or eliminate the need to write about the topic in the first place. As a result, I’ve rewritten large portions of both this book and Fiddler itself. It’s been a slow process, but both projects have benefitted.

Publication of this book will roughly coincide with the release of Fiddler version 2.4.0.0 in the early summer of 2012. If you’re using a later version of Fiddler, you will find some minor differences, but the core concepts will remain the same.

This book is deliberately limited in scope—it covers nearly every aspect of Fiddler and FiddlerCore, but it is not a tutorial on HTTP, SSL, HTML, Web Services or the myriad other topics you may want to understand to fully exploit Fiddler’s feature set. If you want a deeper understanding of web protocols, I can recommend the references I consulted during the development of Fiddler:

- HTTP: The Definitive Guide by David Gourley
- Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement by Balachander Krishnamurthy and Jennifer Rexford
- SSL & TLS Essentials: Securing the Web by Stephen A. Thomas

This book can be read either “straight through” or you can use the Table of Contents and Index to find the topics most interesting to you. Please consider skimming all of the chapters, even those that don’t seem relevant to your needs, because each chapter often contains tips and tricks you might not find elsewhere.

I encourage you to begin by reading the primer in the next chapter, which lays out some terminology and the basic concepts that you’ll need to understand to get the most out of Fiddler and this book.

Enjoy!
**Understanding Icons and Colors**

The default text coloring of each row in the Web Sessions list derives from the HTTP Status (red for errors, yellow for authentication demands), traffic type (*CONNECTs* appear in grey), or response type (CSS in purple, HTML in blue; script in green, images in grey). You can override the font color by setting the Session’s `ui-color` flag from Fiddler-Script.

Each row is also marked with an icon for quick reference as to the Session’s progress, Request type, or Response type:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![up]</td>
<td>The Request is being sent to the server.</td>
</tr>
<tr>
<td>![down]</td>
<td>The Response is being downloaded from the server.</td>
</tr>
<tr>
<td>![pause]</td>
<td>The Request is paused at a breakpoint to allow tampering.</td>
</tr>
<tr>
<td>![pause]</td>
<td>The Response is paused at a breakpoint to allow tampering.</td>
</tr>
<tr>
<td>![head/options]</td>
<td>The Request used the <strong>HEAD</strong> or <strong>OPTIONS</strong> methods, or returned a <strong>HTTP/204</strong> status code. The HEAD and OPTIONS methods allow the client to acquire information about the target URL or server without actually downloading the specified content. The HTTP/204 status code indicates that there is no response body for the specified URL.</td>
</tr>
<tr>
<td>![post]</td>
<td>The Request used the <strong>POST</strong> method to send data to the server.</td>
</tr>
<tr>
<td>![html]</td>
<td>The Response is HTML content.</td>
</tr>
<tr>
<td>![image]</td>
<td>The Response is an image file.</td>
</tr>
<tr>
<td>![script]</td>
<td>The Response is a script file.</td>
</tr>
<tr>
<td>![css]</td>
<td>The Response is a Cascading Style Sheet (CSS) file.</td>
</tr>
<tr>
<td>![xml]</td>
<td>The Response is formatted as Extensible Markup Language (XML).</td>
</tr>
<tr>
<td>![json]</td>
<td>The Response is formatted using JavaScript Object Notation (JSON).</td>
</tr>
<tr>
<td>![audio]</td>
<td>The Response is an audio file.</td>
</tr>
<tr>
<td>![video]</td>
<td>The Response is a video file.</td>
</tr>
<tr>
<td>![silverlight]</td>
<td>The Response is a Silverlight applet.</td>
</tr>
<tr>
<td>![flash]</td>
<td>The Response is a Flash applet.</td>
</tr>
<tr>
<td>![font]</td>
<td>The Response is a font file.</td>
</tr>
<tr>
<td>![content-type]</td>
<td>The Response’s <strong>Content-Type</strong> is not a type for which a more specific icon is available.</td>
</tr>
<tr>
<td>![connect]</td>
<td>The Request used the <strong>CONNECT</strong> method. This method is used to establish a tunnel through which encrypted HTTPS traffic flows.</td>
</tr>
</tbody>
</table>
The Session wraps a HTML5 WebSocket connection.

The Response is a **HTTP/3xx** class redirect.

The Response is a **HTTP/401** or **HTTP/407** demand for client credentials, or a **HTTP/403** error indicating that access was denied.

The Response has a **HTTP/4xx** or **HTTP/5xx** error status code.

The Session was aborted by the client application, Fiddler, or the Server. This commonly occurs when the client browser began downloading of a page, but the user then navigated to a different page. The client browser responds by cancelling all in-progress requests, leading to the Aborted Session state.

The Response is a **HTTP/206** partial response. Such responses are returned as a result of the client performing a **Range** request for only a portion of the file at the target URL.

The Response is a **HTTP/304** status to indicate that the client’s cached copy is fresh.

The Web Session is unlocked, enabling modification after normal session processing has been completed.

**Keyboard Reference**
The following keyboard shortcuts are supported by the Web Sessions list:

<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spacebar</td>
<td>Activate and scroll the currently-focused session into view.</td>
</tr>
<tr>
<td>CTRL+A</td>
<td>Select all sessions.</td>
</tr>
<tr>
<td>ESC</td>
<td>Unselect all sessions.</td>
</tr>
<tr>
<td>CTRL+I</td>
<td>Invert selection; selected sessions are unselected and vice versa.</td>
</tr>
<tr>
<td>CTRL+X</td>
<td>Remove all sessions (subject to the <code>fiddler.ui.CtrlX.KeepMarked</code> preference.)</td>
</tr>
<tr>
<td>Delete</td>
<td>Remove selected sessions.</td>
</tr>
<tr>
<td>Shift+Delete</td>
<td>Remove all unselected sessions.</td>
</tr>
<tr>
<td>R</td>
<td>Replay the current request</td>
</tr>
<tr>
<td>SHIFT+R</td>
<td>Replay the current request multiple times (specified in the subsequent prompt).</td>
</tr>
<tr>
<td>U</td>
<td>Unconditionally replay the current request, sending no <strong>If-Modified-Since</strong> and <strong>If-None-Match</strong> headers.</td>
</tr>
<tr>
<td>SHIFT+U</td>
<td>Unconditionally replay the current request multiple times (the count is specified in the subsequent prompt).</td>
</tr>
<tr>
<td>P</td>
<td>Attempt to select the “parent” request that triggered this request and set focus to it. This feature depends on the HTTP <strong>Referer</strong> header’s value.</td>
</tr>
<tr>
<td>C</td>
<td>Attempt to select all “child” requests that were provoked by this response. This feature depends on the HTTP <strong>Referer</strong> header’s value or the <strong>Location</strong> header on a redirect.</td>
</tr>
</tbody>
</table>
The Fiddler toolbar provides quick access to popular commands and settings.

The buttons and their functions are:

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>Click to add a Comment to all selected Sessions. The comment appears in a column of the Web Sessions list.</td>
</tr>
<tr>
<td>Replay</td>
<td>Click to reissue the selected requests to the server again. Hold the CTRL key while clicking to reissue the requests without any Conditional Request headers (e.g. If-Modified-Since and If-None-Match). Hold the SHIFT key while clicking to be prompted to specify the number of times each request should be reissued.</td>
</tr>
<tr>
<td>Remove</td>
<td>Shows a menu of options for removing Sessions from the Web Sessions list:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Remove all</strong> removes all Sessions from the list.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Images</strong> removes all Sessions that returned an image.</td>
</tr>
<tr>
<td></td>
<td>- <strong>CONNECTs</strong> removes all CONNECT tunnels.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Non-200s</strong> removes all non-HTTP/200 responses.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Non-Browser</strong> removes all requests that were not issued by a web browser.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Complete and Unmarked</strong> removes Sessions which are in the Done or Aborted state and which are unmarked and have no Comment set.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Duplicate response bodies</strong> removes any Session which has no response body or has a response body which was received in an earlier Session in the list.</td>
</tr>
<tr>
<td>Resume</td>
<td>Resumes all sessions which are currently paused at a Request or Response breakpoint.</td>
</tr>
<tr>
<td>Stream</td>
<td>Enable the Stream toggle to disable response buffering for all responses except those for which a breakpoint was set.</td>
</tr>
<tr>
<td>Decode</td>
<td>Enable the Decode toggle to remove all HTTP Content and Transfer encodings from requests and responses.</td>
</tr>
<tr>
<td>Keep: value</td>
<td>The Keep dropdown controls how many Sessions are stored in the Web Sessions list. When the count is reached, Fiddler will begin removing older Sessions to attempt to limit the list to the desired value. Incomplete Sessions and those with comments, markers, or open Inspector windows are not removed.</td>
</tr>
<tr>
<td>Process Filter</td>
<td>Drag and drop the Process Filter icon to an application to create a Filter which hides all traffic except for that which originates from the selected process. Right-click the Process Filter icon to clear a previously set filter.</td>
</tr>
<tr>
<td>Find</td>
<td>Opens the Find Sessions window.</td>
</tr>
<tr>
<td>Save</td>
<td>Saves all Sessions to a SAZ file.</td>
</tr>
<tr>
<td>Camera</td>
<td>Adds a JPEG-formatted screenshot of the current desktop to the Web Sessions list.</td>
</tr>
<tr>
<td>Browse</td>
<td>If one session is selected, opens Internet Explorer to the target URL. If zero or multiple Sessions are selected, opens Internet Explorer to about:blank.</td>
</tr>
</tbody>
</table>
EXTENDING FIDDLER’S UI - ADDING COLUMNS TO THE WEB SESSIONS LIST

FiddlerScript can also be used to add new columns to the Web Sessions list, either by using attributes or by making a method call.

Binding Columns using Attributes

The `BindUIColumn` attribute is used to create a new column in the Web Sessions list and bind to it a method in the script that will calculate the text for that column. The method must accept a `Session` object as a parameter, and return a `string` as its result.

The following script adds a new column to the Web Sessions list that shows the HTTP Method for each Session:

```java
BindUIColumn("Method", 60)
public static function FillMethodColumn(oS: Session) {
    if ((oS.oRequest != null) && (oS.oRequest.headers != null))
    {
        return oS.oRequest.headers.HTTPMethod;
    }
    return String.Empty;
}
```

After this function is added to the script, a new `Method` column is added to the UI and values are added to the column for each subsequent Session:

```
<table>
<thead>
<tr>
<th>#</th>
<th>Method</th>
<th>Protocol</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>GET</td>
<td>HTTP</td>
<td><a href="http://www.fiddler2.com">www.fiddler2.com</a></td>
</tr>
<tr>
<td>27</td>
<td>CONNECT</td>
<td>HTTP</td>
<td>Tunnel to</td>
</tr>
<tr>
<td>28</td>
<td>CONNECT</td>
<td>HTTP</td>
<td>Tunnel to</td>
</tr>
<tr>
<td>29</td>
<td>POST</td>
<td>HTTPS</td>
<td>beta.urs.microsoft.com</td>
</tr>
<tr>
<td>30</td>
<td>POST</td>
<td>HTTPS</td>
<td>beta.urs.microsoft.com</td>
</tr>
<tr>
<td>31</td>
<td>GET</td>
<td>HTTP</td>
<td><a href="http://www.fiddler2.com">www.fiddler2.com</a></td>
</tr>
</tbody>
</table>
```

Your method must be robust against being called before the data it relies upon is ready. For instance, if you were to add a column that counts the number of times the word `fuzzle` appears in the HTTP response, your method should immediately return an empty string every time it is called until the responseBodyBytes array is created after the response is read from the server. Otherwise, the method will throw a Null Reference Exception every time it is called before the server response is completed.

Because your function will run multiple times for each Session as the Session proceeds from one state to the next, you should ensure that it runs as quickly as possible. One strategy to minimize the work of this function is to cache values...
BUILDING EXTENSION INSTALLERS

You may install your extensions using any technology you like. Fiddler simply requires that its Assembly .dll appear in the correct folder to load it next time that Fiddler launches.

Fiddler and all of the extensions I’ve written are installed using setup programs built using the Nullsoft Scriptable Install System (NSIS). You can get this great freeware from http://nsis.sourceforge.net/Download. NSIS allows you to write a script that is compiled into a compressed executable file containing all of the binaries that make up your project. The resulting setup program is small and works properly across all versions of Windows.

The only significant shortcoming I’ve encountered with NSIS is that it does not support Unicode, so you may need to use a different technology like WIX (http://wix.sourceforge.net/) if you want your installer to use non-Latin characters (e.g. Japanese).

A full explanation of how to use NSIS is beyond the scope of this book—the tool’s website offers plenty of documentation at http://nsis.sourceforge.net/Docs/. However, I’ll share an example setup script you can use to get started.

; In a NSIS Script, the semi-colon is a comment operator
Name "MyExtension"

; TODO: Set a specific name for your installer’s executable
OutFile "InstallMyExtension.exe"
; Point to an icon to use for the installer, or omit to use the default
Icon "C:\src\MyExt\MyExt.ico"
XPStyle on  ; Enable visual-styling for a prettier UI

; Explicitly demand admin permissions because we're going to write to
; Program Files. This prevents the "Program Compatibility Assistant" dialog.
; Note, you can use "user" here if you'd like, but then you must only write
; to HKCU and per-user writable locations on disk.
RequestExecutionLevel "admin"

; Maximize compression
SetCompressor /solid lzma

BrandingText "v1.0.1.0"  ; Text shown at the bottom of the Setup window

; TODO: Set the install directory to the proper folder.
;
; To install to the Extensions folder, use:
InstallDir "$PROGRAMFILES\Fiddler2\Scripts\"
InstallDirRegKey HKLM "SOFTWARE\Microsoft\Fiddler2" "LMScriptPath"

; To install to the Inspectors folder, use:
;InstallDir "$PROGRAMFILES\Fiddler2\Inspectors\"
Inspecting the Session Object

In the original Inspectors API, the Inspectors were never provided a reference to the Session object under Inspection-- only the headers and body would be provided. This provided for a simple, easily understood API contract, but this simplicity presented a number of shortcomings. For instance, it was impossible for an Inspector to get or set flags on the Session object, and even examining properties of the Session was impossible. For instance, the Caching Response Inspector was unable to determine whether the inspected traffic used HTTPS because the URL (and thus the protocol scheme) only appears in the request headers, which were never available to a Response Inspector.

To resolve these shortcomings, the Inspector2 base class was augmented with four additional virtual methods:

```csharp
public virtual void AssignSession(Session oS)
public virtual bool CommitAnyChanges(Session oS)
public virtual bool UnsetDirtyFlag()
public virtual InspectorFlags GetFlags()
```

These methods allow an Inspector to be passed a Session object rather than having individual header and body properties set using the IRequestInspector2 and IResponseInspector2 interfaces. If your Inspector does not override these virtual methods, Fiddler will simply access the headers and body properties on the interface, and you need not implement any of the four virtual methods. If your Inspector does override the AssignSession method, it must still implement all of the legacy properties because not all codepaths in Fiddler call the newer virtual methods. Specifically, when editing a response using the AutoResponder tab, no Session object is available, so the legacy properties will be used.

The AssignSession method is called when the user selects a session in the Web Sessions list when your Inspector’s tab is visible. In your overridden method, your Inspector should update its UI based on the headers and/or body of the session. Note that your Inspector must itself examine the Session’s state to determine whether the Inspector should be readonly, as shown in the following snippet:

```csharp
public override void AssignSession(Session oSession)
{
    if ((null == oSession) || !oSession.bHasResponse)
    {
        Clear();
        return;
    }

    UpdateUIFromHeaders(oSession.oResponse.headers);
    UpdateUIFromBody(oSession.responseBodyBytes);

    bool bIsReadOnly = ((oSession.state != SessionStates.HandTamperResponse) && !oSession.oFlags.ContainsKey("x-Unlocked"));
    UpdateReadOnlyState(bIsReadOnly);
}
```
As you’ve seen in prior chapters, you can extend Fiddler’s functionality with both script and .NET code, and this is the best approach for building new functionality for most users. However, in some scenarios, like test automation, it would be more natural to add proxy functionality into an existing tool or test harness instead of using the entirety of Fiddler for the job.

Enter **FiddlerCore**. FiddlerCore is a class library that you can reference in your .NET applications to add Fiddler-like proxy functionality to .NET programs with none of the Fiddler user-interface. This diagram shows the difference between extending Fiddler with your code and extending your code with FiddlerCore:

If you’ve previously built a Fiddler extension, you’ll find that programming against FiddlerCore is an easy adjustment. Many FiddlerCore-based applications are first prototyped as a Fiddler extension before being moved into a standalone program. Building your code on Fiddler first allows you to easily see what is happening to web traffic using Fiddler’s Inspectors. Once you’re using FiddlerCore, you can only see the web traffic by adding logging functionality to your application (unless you chain your FiddlerCore application to an upstream or downstream Fiddler instance!).

---

**Overview**

As you’ve seen in prior chapters, you can extend Fiddler’s functionality with both script and .NET code, and this is the best approach for building new functionality for most users. However, in some scenarios, like test automation, it would be more natural to add proxy functionality into an existing tool or test harness instead of using the entirety of Fiddler for the job.

Enter **FiddlerCore**. FiddlerCore is a class library that you can reference in your .NET applications to add Fiddler-like proxy functionality to .NET programs with none of the Fiddler user-interface. This diagram shows the difference between extending Fiddler with your code and extending your code with FiddlerCore:

If you’ve previously built a Fiddler extension, you’ll find that programming against FiddlerCore is an easy adjustment. Many FiddlerCore-based applications are first prototyped as a Fiddler extension before being moved into a standalone program. Building your code on Fiddler first allows you to easily see what is happening to web traffic using Fiddler’s Inspectors. Once you’re using FiddlerCore, you can only see the web traffic by adding logging functionality to your application (unless you chain your FiddlerCore application to an upstream or downstream Fiddler instance!).

---

If you’ve previously built a Fiddler extension, you’ll find that programming against FiddlerCore is an easy adjustment. Many FiddlerCore-based applications are first prototyped as a Fiddler extension before being moved into a standalone program. Building your code on Fiddler first allows you to easily see what is happening to web traffic using Fiddler’s Inspectors. Once you’re using FiddlerCore, you can only see the web traffic by adding logging functionality to your application (unless you chain your FiddlerCore application to an upstream or downstream Fiddler instance!).
Appendix C: Session Flags

A `StringDictionary` field in each Session object contains flags that control the processing or display of the session.

Some flags are set by Fiddler itself, but most are set script or extensions. The list of supported flags grows with each update to Fiddler, and extensions may use their own flags (which have no meaning to Fiddler) to add state information to a given Session.

The flags can be accessed by `oSession.oFlags["flagname"]` or by using the default indexer on the Session object: `oSession["flagname"]`.

Flag names are case-insensitive strings, and most flag values are interpreted case-insensitively. Most of Fiddler’s flags are simply checked for their existence, such that setting any value (even misleading strings like `0`, `false`, and `heck no!`), enables the named behavior. To disable a flag, remove the flag from the Session like so:

```csharp
oSession.oFlags.Remove("flagname");
```

Because most flags are simply tested for existence, a best practice is to use the flag’s value to store a terse explanation of why the flag was set. For instance:

```csharp
oSession["ui-hide"] = "hidden by Hide Images rule";
```

You can view a Session’s flags by using the Properties item on the Web Sessions list’s context menu.

Session Display Flags

The following flags control how a session appears within the Web Sessions list.

<table>
<thead>
<tr>
<th>Flag Name</th>
<th>ui-hide</th>
</tr>
</thead>
</table>
| **Explanation** | The session will not appear within the Web Sessions list. One of the most commonly used flags, `ui-hide` is used by script or extensions to avoid cluttering the Web Sessions list with uninteresting traffic.
| **Supported Values** | Any value will hide the Session.

Typically, you should provide a terse explanation of why the Session was hidden, so that if the user activates the Troubleshoot Filters feature on the Help menu, the UI will explain why the Session was hidden.

By default, Fiddler will not hide requests that it itself generated (e.g. using the Composer). However, if the `ui-hide` flag’s value contains the word `stealth`, the Session will be hidden unconditionally.