開発者必読！

# “もしも”のときのデバッグ技法

.NETアプリケーション障害解析

株式会社NTTデータ
飯山 教史
*IIYAMA, Takashi*

第 11 回

## HTTPモジュールでWebアプリケーションのログをとる

### ログの有用性

デバッグ時に限らず、Webアプリケーションでログを記録しておくことはとても重要である。デバッグであれば、障害が起こったときにどのURLがアクセスされたかを記録し、そのときのパラメータを記録するなどの用途が考えられる。デバッグでなくても、たとえばRefererヘッダーの情報を記録していればどのURLからリンクされているのかを監視でき、これをマーケティングで利用すればユーザーの嗜好などが把握できるだろう。

このように有用なログ機能だが、IISが標準で提供しているロギング機能は拡張性に乏しく有用な情報を記録することができない場合がある。

そこで今回はASP.NETでログ機能を独自に開発する方法として、数多く存在する手法の中から「HTTPモジュールを開発する方法」を説明する。

**Level**

1 2 3 4 (5)

**Technology Tools**

- ◯ Visual Basic
- ✓ Visual C#
- ◯ Visual C++
- ◯ SQL Server
- ◯ Oracle
- ◯ Access
- ◯ ASP.NET
- ✓ Other:
  ↓
  Visual Studio .NET 2003

### イベントハンドラとは？

HTTPモジュールを利用して独自のログ機能を作成するには、「イベント」を理解しなくてはならない。イベントは「クラスで“外部に発生を通知する”と定義されたアクションの発生を外部に送信する仕組み」と言える。アクションにはさまざまなパターンがあるが、一番イメージしやすいのはVBプログラマならおなじみのボタンを押したら実行されるイベントハンドラである。これは、まさに「ボタンを押す」というアクションをイベントハンドラに通知しそのコードを実行するという意味で、アクションの発生を外部に通達する仕組みとなっている（図1）。

当然のことだが、イベントを通信する送信元のクラスはどのクラスがイベントを受信するかについての情報を何も持っていない。実装時にそれらを予測するのは不可能だからだ。このギャップを埋めるため、送信元と受信元で共有できる情報をもとにイベント発生の情報をやり取りする。この機能は.NET Frameworkでは「Delegate（デリゲート）」と呼ばれている。このクラスはメソッドへの参照を格納（イベントハンドラ的な挙動をする）できるので、コールバック関数のような単なるポインタではなくタイプセーフなアクセスを保障できるのである[注1]。

HTTPモジュールはこのイベント／デリゲートの機能を利用してASP.NETの処理をフックし、自前で定義したログ

図1：イベントハンドラの仕組み

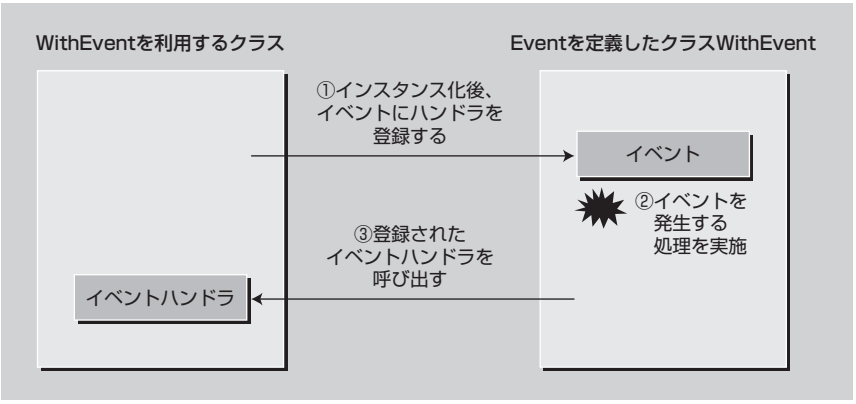



表1：システムに組み込まれているHTTPモジュール

| |
|---|
| OutputCache |
| Session |
| WindowsAuthentication |
| FormsAuthentication |
| PassportAuthentication |
| UrlAuthorization |
| FileAuthorization |
| ErrorHandlerModule |
| HttpModuleSample |
| DefaultAuthentication |

を記録できるようになる。具体的には、ASP.NETで定義された各種イベントに対するデリゲートをHTTPモジュールとして実装し、Web.configファイルで宣言してシステムに組み込むことになる。

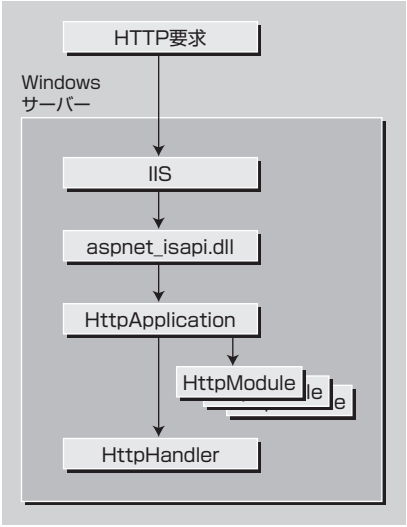
HTTPモジュールの本来の役割を理解するため、以降ではASP.NETのアーキテクチャとしてHTTPパイプラインがどのように動いているのかを説明する。

## HTTPリクエストの処理

HTTP要求がポート80（HTTPSとSSLの場合は443）からあった後、最初にWebサーバーがこの要求を処理する。Internet Information Services（IIS）はHTTPリクエストを受け取ると、URLをチェックして最終要求先を確認する。この振り分けには拡張子が利用され、「.aspx」であればその処理をaspnet_isapi.dllにマップする。このDLLはHTTPリクエストを受け取るとASP.NETのワーカープロセスにあたるaspnet_wp.exeへ引き渡し、このEXEの実態であるプロセスによってHTTP要求は処理されるのである。

このaspnet_wp.exeの中ではHTTPリクエストはHttpContextクラスのインスタンスにラップされる[注2]。この状態でHTTPリクエストはHttpApplicationのインスタンスを通るが、その間にHTTPリクエストを処理するためにHttpApplicationクラスがHTTPモジュールを呼び出し、HTTPリクエストの処理を進めるのである。そのためHTTPモジュールで定義した処理が呼び出されるきっかけとなるイベントはこのHttpApplicationクラスが持っており、事実、システムレベルのHTTPモジュール（表1）は認証や出力のキャッシングなどの機能を実装しているが、それらはHttpApplicationから呼び出されているのである（図2）[注3]。

図2：HTTPリクエストの処理シーケンス




HTTPリクエストを処理する最後のクラスはHttpHandlerである。System.Web.UI.Pageクラスは、IHttpHandlerインターフェイスを実装しているHttpHandlerクラスであり、HTTPリクエストはこのPageクラスで処理が終了するのである。

注1）イベント／デリゲートについて説明するとそれだけで一回分の連載が終わってしまうので、詳細は以下のURLを見てもらいたい。
http://www.microsoft.com/japan/msdn/library/default.asp?url=/japan/msdn/library/ja/cpguide/html/cpconeventsdelegates.asp

注2）HttpContextクラスにはResponseとRequestに関する情報が含まれているので、ログをとるときにはこのクラスにアクセスする。

注3）動作中のHTTPモジュールのリストを作りたいときにはHttpApplicationのModulesプロパティを利用する。