# Webアプリケーションを支える「状態」の概念

## あなたに最適なページ移動のノウハウ

山田 祥寛
*YAMADA, Yoshihiro*

**Technology Tools**

- ☑ Visual Basic .NET
- ❑ Visual C# .NET
- ❑ SQL Server 2000
- ❑ Oracle 9i
- ❑ Access 2002
- ☑ ASP.NET
- ☑ Internet Information Services
- ☑ Other:
  MSDE 2000

**Level**

★ ★ ☆ ☆ ☆

**Samples**

### アプリケーションとしてのWebの悩み

HTTPという、ステートレス（状態を管理しない）な通信プロトコルをベースにおくWebにおいて、最大の悩みとはページ間にまたがるデータの保持にほかなりません。

HTTPにおいては、クライアントがリクエストし、サーバーがレスポンスするそのやりとりが1回の「完結した」通信であり、それ以上のものでもそれ以下のものでもありません。たとえば、ページAでログインを行なったとしても、別なページBでは「ログインした」という事実を自動的に知ることはできないのです。

開発者はこれまで異なるページ間で情報を共有するために、クエリ情報、隠しフィールド（hidden）、クッキー、セッションと、さまざまな手法を考案してきたのです（図1）。Webアプリケーションの歴史とは、ある意味「状態」を如何に管理するかの歴史であったともいえます。

### イベントドリブンと「状態管理」の密接な関係

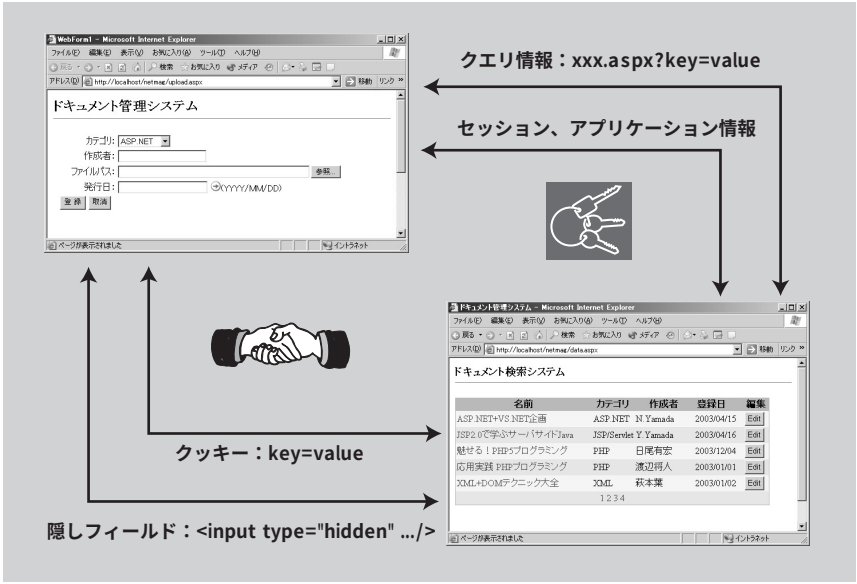
ASP.NETは「サーバーラウンドトリップ」あるいは「ポストバック」と呼ばれるクライアント／サーバー間の密なデータのやりとりを前提としたイベントドリブンのモデルを提供します。ASP.NETにおいて、イベントの発生とはクライアントとサーバー間で通信が発生する、もっといえば、ページがいったんクリアされることと等価でもあります。

つまり、もしもASP.NETが「状態」を管理するための手段を提供していないとしたら、開発者はイベントプロシージャの中に、現在のコントロールの状態を保持するための多くのコードを記述しなければならなかったはずです。

にもかかわらず、ASP.NETではイベント処理が行なわれた後も自然とページの状態が保持されていたのは、状態を管理するための「ビューステート（ViewState）」というしくみをASP.NETが提供していたためです。

「ビューステート」といっても特別なものはありません。試しに、これま

図1：状態管理のさまざまな手法




でに作成したASP.NETページを実行し、ブラウザ上から［ソースの表示］を選択してみてください。コードの中に、以下のような記述が見つかるはずです。

```
<input type="hidden"
name="__VIEWSTATE"
value="dDw5MDkxNjE0N..." />
```

これが「ビューステート」の正体です。

ASP.NETでは、ページ内に配置されたコントロールの状態を「隠しフィールド」という形でセットし、リクエストのつどサーバーに送信することで、状態を保持することを可能にしています。ビューステートの値はハッシュされ、圧縮され、Unicode実装用に暗号化されていますので、効率的に、かつセキュアに情報を交換することができます。

これまでIsPostBackメソッドを介して、初回ロード時の処理とポストバック時の処理とを分岐していましたが、この意味とは実は、

①ポストバック時に展開されたビューステートの内容をPage_Loadプロシージャの内容が上書きしてしまうことを防ぐ
②ビューステートによってデータが展開されているにもかかわらず、Page_Loadイベントで重複して処理を行なうことの無駄を省く

という2点にあったのです。

## ビューステート使用時の留意点

ビューステートは、開発者がほとんど意識する必要もなく、また、クライアントの設定にも依存しないことから、きわめて有用な「状態管理」の手法ではあります。しかし、必ずしも問題がないわけではないことも、開発者としては充分に理解しておくべきでしょう。

以下に、ビューステートの大きな問題点とその対応策を挙げておくことにします。

### ■ページの複雑度によっては大量のPOSTデータが発生する

先にも述べたように、ビューステートの正体は暗号化された隠しフィールドです。つまり、ページが複雑になり、配置されたサーバーコントロールの数が多くなった場合（あるいはDataGridのような巨大なデータを伴うコントロールが配置された場合）、ビューステートのサイズは決して馬鹿にならないものとなります。特に、ダイヤルアップ接続などのナローバンド環境にあるクライアントでは、ビューステートに起因するパフォーマンスの遅延を懸念するべきでしょう。

これに対応する策としては、各サーバーコントロールに共通で用意されたEnableViewStateプロパティの利用が挙げられます。EnableViewStateプロパティは、コントロールが現在の状態を保持するかどうかをTrue/Falseで設定します。デフォルトはTrue（ビューステートを保持する）に設定されていますが、状態を保持する必要のないコントロールについては、EnableViewStateプロパティをFalseに設定しておくことで、ビューステートのサイズを縮減することが可能です。

EnableViewState属性は@Pageディレクティブでも指定することができ、ページ全体におけるビューステートの有効／無効を切り替えることも可能です。