웹디자이너를 위한 ASP 게시판 프로그래밍 1

ASP 내장 객체와 카운터 만들기

- 디지털디자인 http://www.ddcom.co.kr
- 서울시 강남구 역삼동 707-1 두꺼비 빌딩 1102호
- 메일: webmaster@ddcom.co.kr
- Tel. 02-538-0183~4

Digital Design

목 차

이 책에 사용된 예제 파일은 디지털디자인 홈페이지(http://ddcom. co.kr/)의 웹 저작 소스 자료실에서 다운로드 할 수 있습니다. 아래 주소를 클릭하십시오.

http://ddcom.co.kr/ddcompds/view.asp?no=1143&page=1&forum_id=source

웹프로그래밍과 ASP

chapter 1

웹프로그래밍 언어와 그 언어로 만들어진 웹애플리케이션에 대해 알아보겠습니다. 이 책에서는 웹프로그래밍 언어 중에서 특히 ASP을 중심으로 진행하겠습니다.

많은 사람들이 웹서핑을 즐기는 한편, 자신만의 홈페이지를 가지고 싶어합니다. 그래서 HTML도 배우고, 나모나 드림위버 같은 웹에디터도 배우고, 포토샵 등의 그래픽 편집 프로그램도 배웁니다. 그러나 이런 과정은 홈페이지를 만들기 위한 기초 과정일 뿐, 정작 홈페이지를 만들려고 컴퓨터 앞에 앉으면 그다지 큰 도움이 못됩니다. 예를 들어, 자신이 만들려고 하는 홈페이지에서 회원 가입을 받고 한 달에 한 번 정도 정기적인 이메일을 발송하려 할 경우, 이러한 것(이를 웹프로그램이라고 합니다)을 어떻게 만들어야 할지 무척이나 난감할 것입니다.

카운터, 방명록, 게시판, 채팅, 쇼핑몰에 이르기까지 웹에서 자주 접하는 것들은 과연 어떻게 만들어야 할까요? 많은 웹디자이너들이 여기서 주저앉는 경우가 많습니다. 비밀은 바로 웹프로그래밍 언어에 있습니다. 그동안 그저 어렵다고 지나쳐 온 프로그래밍 언어들이 이것들을 해결하는 방법입니다.

웹프로그래밍 언어와 그 언어로 만들어진 웹애플리케이션에 대해 알아봅니다.

01 웹프로그래밍과 웹애플리케이션

웹프로그래밍이란 ASP 같은 웹프로그래밍 언어를 사용해서 웹페이지를 제작하는 것을 말합니다. 여기서 ASP란 Active Server Page의 약자로, 마이크로소프트사에서 개발한 웹애플리케이션 개발을 위한 프로그래밍 언어입니다.

ASP 기본 활용

웹페이지를 만드는 가장 기초적인 프로그래밍 언어는 HTML입니다. 이것은 웹에 종사하는 사람이라면 누구나 한번쯤은 보고 들었을 것이고, 실제로 홈페이지를 만들 때 사용해 보기도 했을 것입니다.

HTML을 한마디로 표현한다면 멍텅구리 언어라고 할 수 있습니다. 단지 웹에서 보여주는 것만 만들 수 있을 뿐, 웹에 접속한 유저와 어떤 커뮤니케이션도 할 수 없습니다. 그래서 많은 사람들이 커뮤니케이션이 가능한 수단을 원했고, 그 방안으로 개발된 것이 바로 웹애플리케이션 프로그래밍 언어입니다.

이렇게 개발된 웹애플리케이션 프로그래밍 언어는 ASP, JSP, PHP, Perl 등 매우 다양합니다. 이 중에서 많은 사람들이 비교적 쉽게 배울 수 있는 것이 ASP입니다. ASP는 주로 윈도우 프로그램을 제작하는 데 사용되는 비주얼베이직(Visual Basic)에 뿌리를 두고 있기 때문에 윈도우 환경의 웹애플리케이션을 제작하는 데 최적의 효과를 발휘합니다.

ASP 같은 웹애플리케이션 프로그래밍 언어의 특징은 웹서버에서 실행된다는 점입니다. 기존의 HTML 언어가 웹브라우저에서 읽혀지고 실행된 후, 그 결과를 웹브라우저 화면에 표시한 데에 반해, ASP는 웹서버에서 실행되고 그 결과값만 웹브라우저에 전달해 화면에 표시합니다.

다음의 코드를 보면 일반적인 HTML 문서처럼 보이지만, 중간에 〈% ~%〉 태그가 포함된 것을 볼 수 있습니다. 〈%~%〉 태그는 ASP 프로그래밍 코드로 HTML 문서 내에 삽입이 가능합니다. 이처럼 ASP 코드는 반드시 〈%로 시작됩니다.

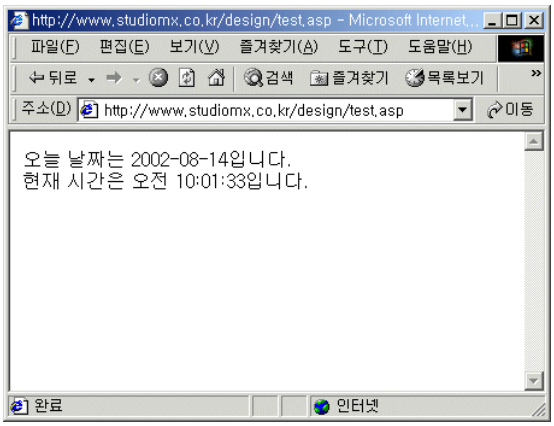
예제 1-1
ASP 코드 예

```
<html>
<head>
</head>
```

```
<body>
오늘 날짜는 <% Response.Write date() %>입니다.
<br>
현재 시간은 <% Response.Write time() %>입니다.
</body>
</html>
```

HTML 문서에 들어 있는 〈% ~ %〉 태그는 웹서버에서 실행된 후, 각각 시스템의 오늘 날짜와 현재 시간을 받아와서 웹브라우저에 전송하게 됩니다. 다음 그림은 위의 코드가 웹브라우저 화면에 표시된 그림입니다.

그림 1-1
[예제 1-1]가 웹브라우저에 표시된 화면

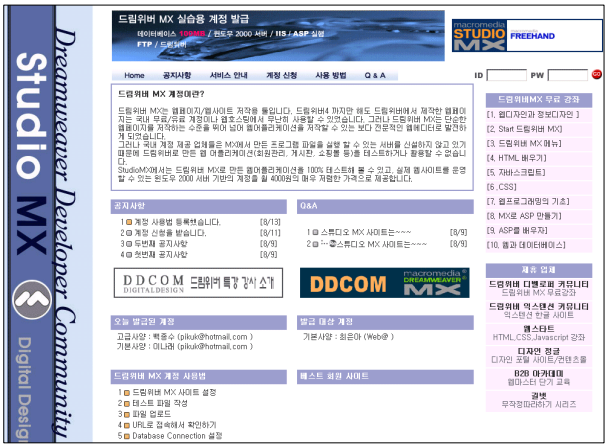


ASP 등으로 만들어진 웹애플리케이션 프로그램은 로컬 작업 환경에서 웹브라우저로 볼 수 없습니다. HTML로 작성된 파일은 로컬 작업 환경에서 웹브라우저에서 해석되기 때문에 바로 확인할 수가 있지만, 웹애플리케이션 프로그램은 웹서버에서 실행되기 때문에 웹브라우저만 있다고 해서 결과를 볼 수 있는 것은 아닙니다.

웹애플리케이션 프로그램은 해당 언어를 지원하는 웹서버에 파일을 업로드한 후에 웹브라우저로 접속해야 볼 수 있습니다. 따라서 HTML 작업을 할 때보다 조금은 복잡하고 번거로워질 수밖에 없습니다.

웹 호스팅

많은 사람들이 홈페이지를 출판하기 위해 포탈 사이트에서 제공하는 무료 홈페이지 계정을 사용합니다. 그러나 대부분의 무료 홈페이지 계정은 웹애플리케이션을 실행할 수 있는 조건을 갖추고 있지 않습니다. 그래서 호스팅 업체에 의뢰한다거나, 직접 웹서버를 구성하는 경우가 많죠. 웹애플리케이션이 실행되는 계정은 필수적인 것입니다. http://www.studiomx.co.kr/home 사이트에서 저렴한 가격으로 웹애플리케이션 계정을 제공하고 있습니다.

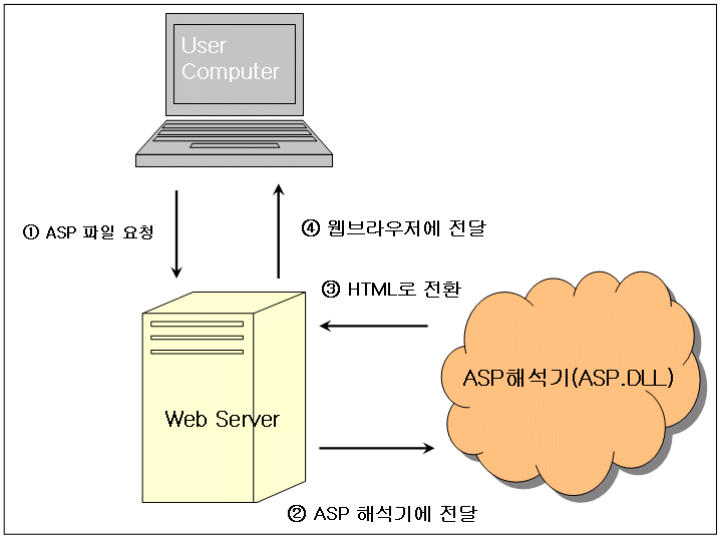


웹애플리케이션이 실행되는 과정

그럼 웹애플리케이션 프로그래밍이 실행되는 과정을 알아보겠습니다. 이 과정을 이해하면 웹애플리케이션 프로그램에 대한 두려움이 사라지면서 일반 HTML 웹페이지를 만드는 것처럼 쉽게 생각될 것입니다.

다음 그림은 웹애플리케이션이 실행되는 과정을 도식화한 것입니다.

그림 1-2
웹애플리케이션 실행 과정



1. 사용자의 컴퓨터에서 웹서버에 웹애플리케이션 파일을 요청합니다. 이 과정은 웹브라우저의 주소 입력줄에 웹애플리케이션 프로그램으로 구축된 웹사이트 주소를 입력해 접속하는 과정이라고 보면 됩니다. 예를 들면, 게시판에 등록된 글을 읽기 위해 게시물 항목을 하나 클릭하는 과정입니다.

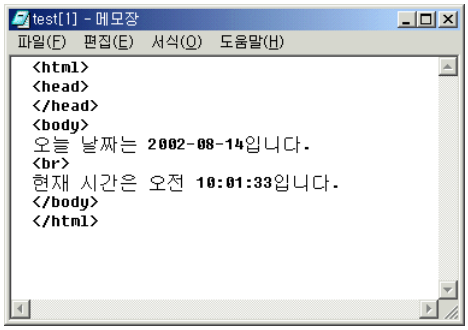
2. 웹브라우저로부터 웹애플리케이션 프로그램 파일을 요청받은 웹서버는 이 파일을 해석하고 실행할 수 있는 프로그램에 파일을 전달합니다. ASP 파일인 경우에는 ASP 해석기인 ASP.DLL이 웹애플리케이션 프로그램 파일을 해석하고 실행합니다.

3. 실행된 웹애플리케이션 프로그램 파일은 HTML 파일로 전환됩니다. 그래야 웹브라우저 화면에서 보일 수 있습니다. 다시 말하지만, 웹브라우저는 HTML 파일만 화면에 표시할 수 있습니다.

4. HTML로 전환된 파일을 웹브라우저가 전송받으면, 웹브라우저는 이를 다시 해석하고 실행한 후, 웹브라우저 화면에 표시합니다.

이상의 4단계 과정을 거쳐 웹애플리케이션 프로그램 파일이 웹브라우저에 표시됩니다. 이 때 웹브라우저에 전송된 파일은 원래 웹서버에 요청한 파일, 즉 ASP에서 만들어진 파일과는 달라집니다. 앞에서 예로 들었던 ASP 파일이 웹브라우저에 표시된 후, 웹브라우저에서 소스 코드를 확인해 보면 다음과 같이 나타납니다.

그림 1-3
메모장으로 [그림 1-1]의 소스코드를 확인한 화면



〈%~%〉 태그로 입력한 ASP 코드가 모두 일반 텍스트로 바뀌어 소스에 표시됩니다. 즉, ASP 코드는 웹서버에서 실행된 후, HTML 문서에 텍스트 형식으로 삽입되는 것입니다.

02 객체지향 웹페이지

객체지향 웹페이지! 말이 너무 생소하죠. 그렇다고 기죽을 필요는 없습니다. 프로그래밍을 할 때에는 어려운 용어를 외우려 하지 말고, 어떤 것이 있는지만 기억해 두기 바랍니다. 언젠가 실제로 프로그래밍할 때 저절로 이해할 수 있을 것입니다.

객체지향 웹페이지란?

객체지향(Object-Oriented) 웹페이지는 쉽게 말해 객체지향 프로그래밍 언어를 사용해서 만들어진다는 것을 의미합니다. ASP, JSP 등이 바로 객체지향 언어입니다. 독자들은 JavaScript라는 언어를 들어봤을 것입니다. 이것은 중간 단계 정도의 객체지향 언어라고 할 수 있습니다.

우리가 볼 수 있는 대부분의 웹페이지는 객체지향 웹페이지로, 대부분 데이터베이스와 연동해 구성됩니다(연동이란 단어를 어렵게 생각할 필요는 없습니다. 연결되어 함께 동작, 즉 데이터를 주고받으면서 움직인다는 뜻입니다). 예를 들어, 쇼핑몰을 생각해 보겠습니다. 1만 개의 상품을 파는 쇼핑몰이 있다고 했을 때, 일반 웹페이지로 만든다면 일단 상품에 대한 세부 설명이 담긴 페이지를 1만 개나 만들어야 합니다. 또한 상품을 목록별로 구별해 놓은 페이지도 만들어야 하고, 상품별 결제 페이지도 만들어야 합니다. 이처럼 기존 HTML로 쇼핑몰 웹페이지를 만들면 무수히 많은 페이지를 만들어야 합니다. 하지만 기본적인 레이아웃이 있고, 내용을 데이터베이스에서 가져와 뿌려주게 만든다면, 상품 설명 페이지는 하나의 파일만 있으면 됩니다. 이것이 바로 객체지향 웹페이지입니다.

객체

객체지향 웹페이지에서는 웹페이지에 사용되는 데이터를 객체화시킵니다. 이렇게 객체가 된 데이터들은 속성 값과 메소드를 갖게 됩니다. 속성은 객체가 가지고 있는 성질이라 생가각하면 되고, 메소드는 객체가 해야 할 일이라 보면 됩니다.

예를 들어, 자동차를 웹페이지로 생각했을 때 자동차에서 사용되는 각종 부품을 객체라고 보면 됩니다. 이 부품 중에서 바퀴는 둥글고, 고무로 되어 있으며, 검정색입니다. 이것이 바로 객체의 속성입니다. 이 바퀴를 떼어내서 다른 자동차에도 사용할 수 있지만, 크기가 다르면 사용할 수 없습니다. 즉, 객체의 속성이 맞아야 호환될 수 있는 것입니다.

다시 말해 웹페이지에서 사용되는 객체는 다른 웹페이지에서 사용될 수도 있고, 그렇지 못할 수도 있는 것입니다. 그리고 메소드는 객체의 행위라고 했는데, 자동차 바퀴는 굴러가야 그 몫을 다하는 것입니다. 이때 바퀴의 메소드로 '굴러가라'는 행위를 지정할 수 있습니다.

객체의 가장 큰 특징은 한 번 만들어 놓으면, 여러 번 사용할 수 있다는 것입니다. 따라서 미리 만들어 놓은 객체를 사용하면 프로그래밍을 빠르고 쉽게 할 수 있습니다.

객체지향 웹페이지 시스템

객체지향 웹페이지의 전체적인 작업 구성을 살펴보면, 가장 먼저 분석을 통한 컨텐츠를 분류합니다. 분류된 컨텐츠는 데이터베이스에 저장되고, 템플릿을 활용한 기본적인 디자인과 인터페이스를 구성합니다. 그 다음으로 여러 웹페이지에 공통으로 들어가는 소스 코드를 파일로 따로 저장하여 웹페이지에 파일 삽입(SSI; Server-Side Include) 형식으로 삽입하게 됩니다. 마지막으로 데이터베이스로부터 저장된 컨텐츠와 필요한 데이터를 불러와서 다른 데이터들과 조합하여 웹페이지를 완성합니다.

결론적으로, 객체지향 웹페이지는 페이지 생성이나 수정시에 불필요한 작업을 줄이고 효율적인 웹사이트를 관리 운영할 수 있게 해줍니다. 예를 들어, 10만권의 책을 파는 대형 서점과 인터넷 서점을 비교해 보면, 대형 서점의 최소 직원이 50명이라고 할 때, 객체지향 웹페이지로 만들어진 인터넷 서점은 한 명의 직원이면 충분합니다.

객체지향 웹페이지를 작성하기 위한 기본적인 시스템은 다음 그림처럼 총 5단계로 요약할 수 있습니다.

1단계

웹페이지를 만들기 전에 보여주고자 하는 컨텐츠가 무엇인지 정확히 파악해야 합니다. 컨텐츠가 정해지면 그에 따른 컨텐츠 분석을 해야 합니다. 분석을 통해 컨텐츠를 카테고리별로 분류하고, 분류된 각 카테고리 조각들을 구조화시켜야 합니다. 예를 들어, 미팅 사이트에서 회원 관리를 위해 성별, 나이, 직업, 사진, 프로필, 연봉, 결혼 유무 등을 도식화하는 것처럼 웹페이지의 내용이 어떤 식으로 분리될 수 있는지 분석해야 합니다.

2단계

1단계에서 웹페이지에 들어갈 내용들을 분리시켰으면, 그 내용이 저장될 데이터베이스 디자인을 합니다. 우선 데이터베이스를 생성하고 필요한 테이블을 추가합니다. 추가된 테이블에 이미 만들어진 도식에 따라 필드를 추가합니다.

3단계

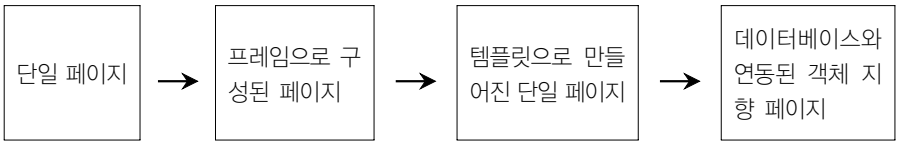
웹페이지의 메뉴나 로그인처럼 모든 페이지에서 공통으로 보여주는 부분은 파일을 따로 저장하여 SSI(Server-Side Include) 형식으로 웹문서에 삽입합니다. 또한 데이터베이스로부터 불러올 데이터를 표시할 페이지 템플릿 디자인을 구성합니다.

4단계

기본적으로 작성된 템플릿 디자인 내에 데이터베이스의 데이터가 표시되어야 합니다. 즉, 하나의 레이아웃으로 여러 웹페이지가 생성된 것처럼 표현하기 위해 SQL 쿼리 문을 사용해서 웹페이지의 적절한 위치에 데이터를 불러옵니다.

5단계

필요에 따라 완성된 템플릿 안에 새로운 객체를 삽입하고, 데이터베이스와의 연동을 통해 최종적인 웹페이지 모습이 만들어 집니다.



ASP의 내장 객체

chapter 2

ASP로 카운터, 방명록, 게시판, 쇼핑몰을 만들기 위해서는 ASP 프로그래밍에서 기본적으로 지원하는 객체를 알아야합니다. 객체에는 프로그램을 처음 개발하는 초급자도 쉽게 사용할 수 있는 내장 객체가 있고, 사용자가 직접 만들어 쓰는 사용자 정의 객체가 있습니다. 먼저 ASP에서 가장 많이 사용되는 다섯 개의 내장 객체에 대해 알아보겠습니다.

01 Request 객체

ASP의 내장 객체 중 첫 번째로 Request 객체에 대해 알아봅니다.

Request 객체는 웹브라우저에 전송된 값(value)을 서버로 가져오기 위해 사용됩니다. 클라이언트 컴퓨터 브라우저의 여러 가지 정보, 사용자들의 요구 등을 담아두는 역할을 하는 것이 바로 Request 객체입니다. 즉, 웹브라우저와 사용자를 나타내는 HTTP 변수, 사용자들이 저장해 두었던 쿠키, URL 뒤에 첨부되어온 정보나 폼 태그에 의해 전달되어지는 정보 등을 얻을 수 있습니다.

Request 객체의 컬렉션, 프라퍼티, 메소드는 다음과 같습니다(알파벳 순으로 정렬함).

컬렉션

ClientCerficate	클라이언트로부터 인증 정보를 읽어들기 위해 인증서에 저장된 값들의 컬렉션
Cookies	사용자 컴퓨터에 저장된 쿠키의 정보를 읽어들이기 위해 HTTP 헤더와 함께 전송되는 쿠키의 값
Form	전송된 폼의 값(Post 방식)
QueryString	전송된 폼의 값(Get 방식: Get 방식은 하이퍼링크 부분으로 전송됨)
ServerVariables	서버에 저장된 정보를 읽어들이기 위한 환경 변수의 값

프라퍼티

TotalBytes	요청이 있을 때 클라이언트가 보내는 바이트 수를 결정

메소드

BinaryRead	Post 요청의 일부로, 서버에 보내는 데이터를 읽어들이는 데 사용

QueryString 컬렉션: Get 방식

Request 객체에서 가장 많이 사용하는 컬렉션(Collection. ASP에서 배열과 비슷한 구조를 가지고 있는 데이터의 구조)이 QueryString과 Form입니다. 이 두 개의 값으로 웹페이지에서 전송된 값을 가져와 서버에서 처리할 수 있습니다.

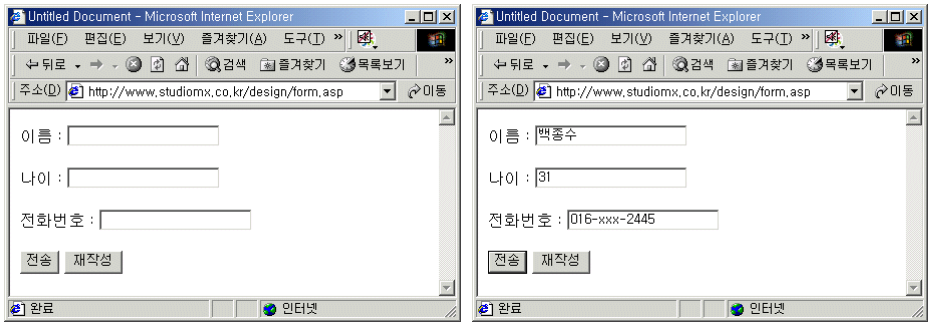
예제 2-1
폼이 삽입된 웹 문서

```
<form method="get" action="request.asp">
  이름 : <input type="text" name="username" size="20"><p>
  나이 : <input type="text" name="age" size="20"><p>
  전화번호 : <input type="text" name="tel" size="20"><p>
  <input type="submit" value="전송">
  <input type="Reset" value="재작성">
</form>
```

[예제 2-1]에서 form 태그의 속성으로 method와 action이 있습니다. method는 정보의 전송 방식을 입력하는데, get과 post 두 가지 방식을 사용합니다. action에는 폼에서 입력받은 정보를 어떤 페이지로 이동할 것인지 경로를 적어 줍니다. 여기서는 method의 두 가지 방식 중에서 먼저 get 방식을 알아보겠습니다.

[예제 2-1]을 웹페이지로 만들어 웹브라우저로 보면 다음 그림처럼 표시됩니다. 여기서 입력란에 내용을 입력하고, [전송] 버튼을 클릭하면 입력된 내용이 〈form〉 태그의 action 속성에서 지정한 된 파일로 전송됩니다. 즉, 백종수, 31, 016-xxx-2445라는 세 개의 텍스트가 request.asp 파일로 전송된다는 것입니다.

그림 2-1
[예제 2-1]을 웹브라우저에서 확인한 화면



이번에는 다른 예제를 만들어 보겠습니다. 이 예제는 [예제 2-1]과 연결된 것입니다.

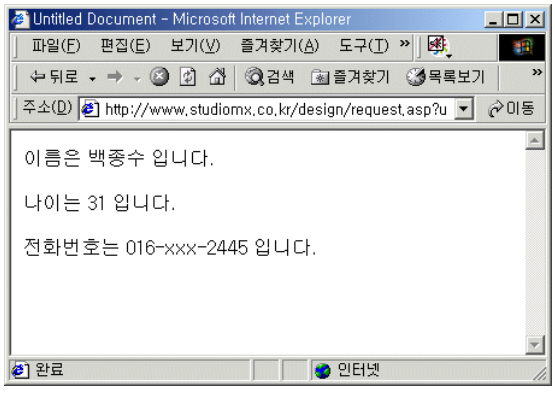
예제 2-2
Request 객체가 삽입된 웹 문서(request.asp)

```
이름은 <%=Request.QueryString("username")%> 입니다.<p>
나이는 <%=Request.QueryString("age")%> 입니다.<p>
전화번호는 <%=Request.QueryString("tel")%> 입니다.<p>
```

[예제 2-2]를 보면 〈% 와 %〉가 나오는데, 이 사이에 들어가는 소스를 ASP 구문이라고 합니다. Request.QueryString("....")은 [예제 2-1]에서 폼으로 전송된 정보를 가져오기 위해 사용된 객체입니다. 폼에 입력된 정보의 이름을 age라고 했을 때, 이 정보를 가져오려면 Request.QueryString("age")라고 입력합니다. 이처럼 get 방식으로 전송되는 정보를 받으려면 QueryString 컬렉션을 사용합니다. 그리고 = 는 정보의 내용을 화면에 출력해 주는 response.write라는 메소드의 다른 표현입니다. 따라서 ASP 코드를 화면에 출력하기 위해서 〈%= ~ %〉라는 구문으로 줄여 사용할 수 있습니다.

[예제 2-1]에서 폼의 각 항목에 입력한 값(이름, 나이, 전화번호)이 각각 백종수, 31, 016-xxx-2445이고, URL 경로가 http://your_domain/form.asp라고 가정해 보겠습니다. [예제 2-2]의 URL 경로가 http://your_ domain/request.asp일 경우, method가 get 방식이라면, [예제 2-2]에서 폼의 내용을 전송 받은 [예제 2-2]는 http://your_domain/request.asp?username=백종수&age=31&tel=016-xxx-2445처럼 표시됩니다. [예제 2-1]에서 전송 내용을 입력하고 [전송] 버튼을 클릭하면 [예제 2-2]가 [그림 2-2]처럼 실행됩니다.

그림 2-2
[예제 2-2] 결과 화면



여기서 method의 get 방식은 앞에서 살펴본 것처럼 폼에 입력된 정보를 URL 뒤에 달고 다닙니다. 이 정보를 꼬리라고도 합니다. 이 꼬리의 길이는 얼마나 될까요? HTTP 프로토콜 스펙에 따라 256바이트~4096바이트까지입니다. 따라서 꼬리에 담을 수 있는 정보량에는 한계가 있습니다. 이 한계를 극복하기 위해 사용되는 방식이 post 방식입니다.

또한 폼의 입력된 정보의 이름만 알면 폼으로 작성된 웹문서가 없어도 URL에 꼬리를 달아서 정보를 전송할 수 있습니다. 웹브라우저의 주소 입력줄에 http://your_domain/request.asp?username=백종수&age=31&tel=016-xxx-2445를 입력해 보면 같은 결과를 얻을 수 있을 것입니다. 따라서 get 방식은 보안에 치명적인 결과를 초래할 수도 있습니다. 이처럼 보안 문제를 해결하려면 post 방식을 사용해야 합니다.

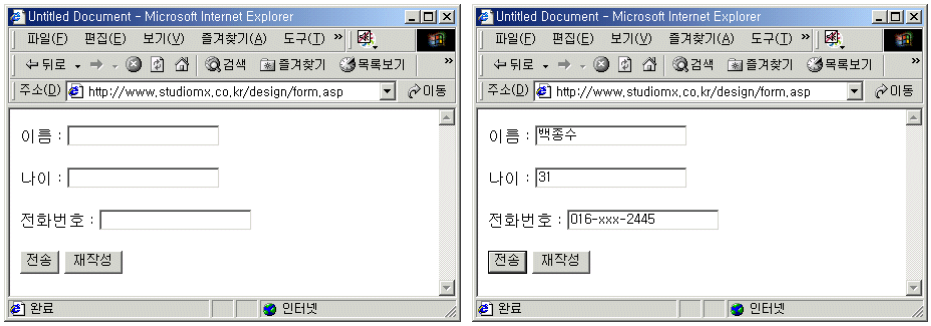
Form 컬렉션: post 방식

다음의 [예제 2-3]과 [예제 2-4]는 post 방식으로 폼에 입력된 정보를 전달하는 예문입니다. 이것은 각각 앞의 get 방식을 설명할 때 살펴 본 [예제 2-1]과 [예제 2-2]에 대응되는 것으로, 결과는 동일합니다.

예제 2-3
폼이 삽입된 웹 문서

```
<form method="post" action="request.asp">
  이름 : <input type="text" name="username" size="20"><p>
  나이 : <input type="text" name="age" size="20"><p>
  전화번호 : <input type="text" name="tel" size="20"><p>
  <input type="submit" value="전송">
  <input type="Reset" value="재작성">
</form>
```

그림 2-3
[예제 2-3] 결과 화면



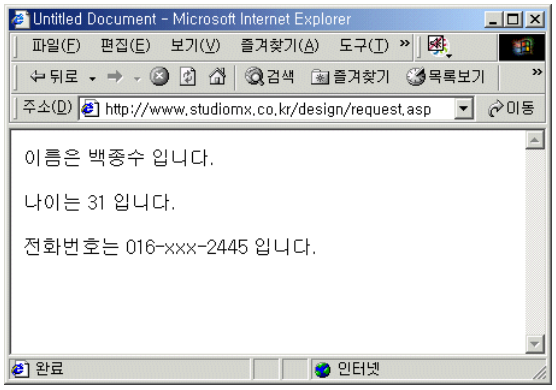
예제 2-4
Request 객체가 삽입된 웹 문서(request.asp)

```
이름은 <%=Request.Form("username")%> 입니다.<p>
나이는 <%=Request.Form("age")%> 입니다.<p>
전화번호는 <%=Request.Form("tel")%> 입니다.<p>
```

[예제 2-3]에서 사용한 <form> 태그의 method가 post로 설정되어 있습니다. post 방식으로 정보를 전달할 경우, 이 정보는 Request.QueryString을 전달받지 못합니다. post 방식으로 전송된 정보를 받으려면 QueryString 컬렉션 대신 Form 컬렉션을 사용해야 합니다. 따라서 정보를 전달 받을 웹 문서에서는 Request.Form을 사용합니다.

여기서 [예제 2-3]의 URL 경로가 http://your_domain/form.asp이고, [예제 2-4]의 URL 경로가 http://your_domain/request.asp일 경우, method가 post 방식이라면, [예제 2-3]에서 폼의 내용을 전송받은 [예제 2-4]는 http://your_domain/request.asp로 표시됩니다. 즉, URL의 꼬리를 달고 다니지 않게 됩니다. 이처럼 URL에 변화 없이 표시되는 이유는 post 방식으로 정보를 전송할 때 HTTP 헤더에 정보를 포함시켜 전송하기 때문입니다. 이렇게 전달된 정보는 get 방식에서의 결과와 똑같습니다. 단지 정보를 전달하는 방식의 차이일 뿐입니다.

그림 2-4
[예제 2-4] 결과 화면



그러나 한 가지 알아두어야 할 것이 있습니다. 폼으로 정보를 전송할 때는 post나 get 방식 중 하나를 선택해서 사용할 수 있지만, 하이퍼링크에서는 선택의 여지없이 get 방식으로 정보를 전달합니다. 이 때 정보를 전달받는 웹 문서에서는 Request.QueryString으로 정보를 받아야 합니다.

ServerVariables 컬렉션

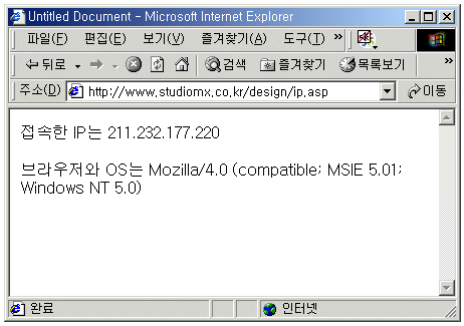
ServerVariables 컬렉션은 서버의 정보와 서버에 접속한 클라이언트의 정보를 얻을 수 있습니다. 이 컬렉션에서는 정보를 얻기 위해 미리 저장되어 있는 여러 종류의 키를 사용합니다.

예를 들어, 웹사이트에 접속한 유저의 IP 주소를 알고 싶다면 REMOTE_HOST라는 키를 사용합니다. 다음 소스는 아래 그림과 같은 결과를 보여 줍니다.

예제 2-5
클라이언트의 정보를 알아내는 소스

```
접속한 IP는 <%=Request.ServerVariables("REMOTE_HOST")%><p>
브라우저와 OS는 <%=Request.ServerVariables("HTTP_USER_AGENT")%><p>
```

그림 2-5
[예제 2-5] 결과 화면



이 구문은 클라이언트의 IP와 웹브라우저 정보를 화면에 표시해 줍니다. 이처럼 Server Variables 컬렉션의 다양한 키를 사용하면 여러 가지 정보를 얻을 수 있습니다. [표 1-1]은 ServerVariables에서 사용하는 키와 해당 정보로, 웹사이트에 접속해 있는 어떤 사용자의 정보입니다.

표 1-1 ServerVariables 컬렉션

키	정보
ALL_HTTP	HTTP_ACCEPT:image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */* HTTP_ACCEPT_LANGUAGE:ko HTTP_CONNECTION:Keep-Alive HTTP_HOST:dart98.bestwebmaster.co.kr HTTP_USER_AGENT:Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0) HTTP_ACCEPT_ENCODING:gzip, deflate
ALL_RAW	Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */* Accept-Language: ko Connection: Keep-Alive Host: dart98.bestwebmaster.co.kr User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0) Accept-Encoding: gzip, deflate
APPL_MD_PATH	/LM/W3SVC/10/Root
APPL_PHYSICAL_PATH	D:\FreeHome\bestwebmaster\dart98₩
AUTH_PASSWORD	
AUTH_TYPE	

AUTH_USER	
CERT_COOKIE	
CERT_FLAGS	
CERT_ISSUER	
CERT_KEYSIZE	
CERT_SECRETKEYSIZE	
CERT_SERIALNUMBER	
CERT_SERVER_ISSUER	
CERT_SERVER_SUBJECT	
CERT_SUBJECT	
CONTENT_LENGTH	0
CONTENT_TYPE	
GATEWAY_INTERFACE	CGI/1.1
HTTPS	off
HTTPS_KEYSIZE	
HTTPS_SECRETKEYSIZE	
HTTPS_SERVER_ISSUER	
HTTPS_SERVER_SUBJECT	
INSTANCE_ID	10
INSTANCE_META_PATH	/LM/W3SVC/10
LOCAL_ADDR	211.63.78.221
LOGON_USER	
PATH_INFO	/servervariable.asp
PATH_TRANSLATED	D:\FreeHome\bestwebmaster\dart98\servervariable.asp
QUERY_STRING	
REMOTE_ADDR	211.63.78.221
REMOTE_HOST	211.63.78.221
REMOTE_USER	
REQUEST_METHOD	GET
SCRIPT_NAME	/servervariable.asp
SERVER_PORT	80
SERVER_PORT_SECURE	0
SERVER_PROTOCOL	HTTP/1.1

SERVER_SOFTWARE	Microsoft-IIS/5.0
URL	/servervariable.asp
HTTP_ACCEPT	image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
HTTP_ACCEPT_LANGUAGE	ko
HTTP_CONNECTION	Keep-Alive
HTTP_HOST	dart98.bestwebmaster.co.kr
HTTP_USER_AGENT	Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)
HTTP_ACCEPT_ENCODING	gzip, deflate

Cookies 컬렉션

쿠키는 유저의 컴퓨터에 유저의 정보를 저장해 놓고 필요할 때마다 사용할 수 있도록 해주는 파일로, 파일의 크기(4Kb)가 쿠키만큼 작기 때문에 붙여진 이름입니다. 쿠키 정보는 서버에 누적되는 것이 아니라 유저의 컴퓨터에 저장됩니다. 그러나 유저의 컴퓨터에 저장된 쿠키를 활용하는 것은 역시 서버입니다.

Request 객체는 Cookies 컬렉션을 사용해서 쿠키를 불러옵니다. 사용자의 쿠키를 불러오려면 다음 구문을 사용합니다.

```
<%=Request.Cookies("user")%>
```

Requset 객체는 쿠키를 불러올 뿐, 유저의 컴퓨터에 저장하는 기능은 없습니다. 쿠키를 유저의 컴퓨터에 저장하기 위한 객체는 다음 절에서 배우는 Response 객체입니다.

02 Response 객체

Request가 서버에 정보를 전송하는 역할을 한다면, Response 객체는 서버의 정보를 사용자의 컴퓨터로 전송해서 저장하는 일을 합니다. Response 객체가 하는 일은 화면 출력 URL 이동, 쿠키 저장 등이 있습니다.

Response 객체의 컬렉션, 프라퍼티, 메소드는 다음과 같습니다.

컬렉션

Cookies	사용자의 컴퓨터에 쿠기 파일 생성

프라퍼티

Buffer	페이지의 버퍼 완료 여부 결정(True/Fasle)
cachecontrol	Proxy Server의 ASP에 의해 만들어진 결과를 캐시에 저장할 것인지의 여부 결정
charset	컨텐트 헤더에 문자셋을 덧붙임
ContentType	HTTP의 컨텐트 타입을 지정
Expires	캐시에 저장할 시간을 분 단위로 지정
ExpiresAbsolute	캐싱된 ASP 페이지가 소멸될 시간 결정
isclientconnected	사용자가 서버에 연결되어 있는지의 여부를 체크
Status	서버에서 리턴한 HTTP의 상태값을 결정

메소드

AddHeader	HTML 헤더에 ASP 페이지 정보 전송
AppendToLog	서버의 Log에 텍스트 추가
BinaryWrite	텍스트를 브라우저에 문자셋없이 출력
Clear	버퍼링된 HTML의 내용을 삭제함으로써 브라우저에 전달되지 않음
End	ASP 페이지의 코드 처리를 중단하고 현재의 결과를 리턴
Flush	버퍼에 저장된 결과를 바로 리턴
Redirect	지정된 URL로 이동
Write	크라이언트에 변수나 문자열을 출력

이것들을 사용되는 분야에 따라 다시 분류해보면 [표 1-2]처럼 정리할 수 있습니다.

표 1-2 Response 객체의 사용 분야별 분류

분류	설명
페이지 출력에 관련	Write, BinaryWrite
페이지 버퍼 관련	buffer, Flush, Clear, End
페이지의 속성 관련	expires, ExpiresAbsolute, ContentType, AddHeader, status
쿠키관련	Cookies
URL 이동관련	redirect
사용자의 연결 체크	isclientconnected

Write 메소드

Response 객체의 write 메소드는 ASP에서 가장 많이 사용되는 메소드입니다. 그 이유는 서버에서 클라이언트로 보내지는 HTML의 출력 부분을 담당하고 있기 때문입니다. 여기서 한 가지 알아둘 점은 "Response.write로 출력한다"는 의미는 웹브라우저의 화면 출력을 의미하는 것이 아니라 HTML 문서 내의 출력을 의미합니다. 즉, HTML에서 일반 텍스트로 전환되어 나타나는 것입니다.

또한 앞에서도 잠깐 언급되었지만, 가장 많이 쓰는 구문이기 때문에 Reponse.write를 줄여서 = 로 대신 사용하기도 합니다.

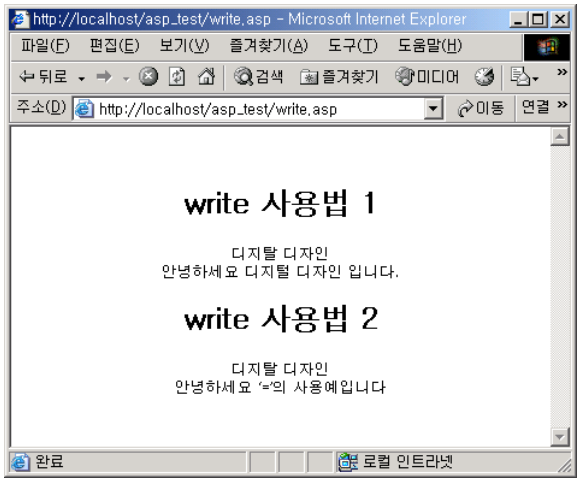
다음은 Write 메소드를 사용한 예입니다.

예제 2-6
Write 메소드 사용 예

```
<%
  str = "디지탈 디자인"
%>
<HTML>
<BODY>
  <P> </P>
  <font face="돋움" size="2"><center>
  <h2>write 사용법 1</h2>
<%
  Response.Write str
  Response.Write "<br>"
```

```
  Response.Write "안녕하세요 디지털 디자인 입니다."
%><p>
  <h2>write 사용법 2</h2>
<%=str%><br>
<%="안녕하세요  '='의 사용예입니다"%>
  </center></font>
</BODY>
</HTML>
```

그림 2-6
[예제 2-6] 결과 화면



Buffer, Flush, Clear, End 메소드

Buffer는 페이지 버퍼에 관련된 메소드로, 사용자가 페이지의 정보를 받는 데에 대한 제어를 할 수 있습니다. Buffer는 true와 false라는 두 가지 불린(boolean) 값을 갖습니다. Buffer가 true로 설정된 페이지는 모든 스크립트를 처리한 후, 웹브라우저에 한꺼번에 전송합니다. 반대로 false로 설정된 페이지는 순서대로 로딩됩니다. 즉, ASP 코드가 ASP 실행기에서 실행되는 순서대로 웹브라우저에 전송한다는 뜻입니다. 일반적으로 ASP로 만들어진 웹페이지는 Buffer=false를 기본으로 로딩합니다.

그리고 Buffer가 true인 경우에는 Flush, Clear, End를 사용할 수 있지만, false이면 Flush, Clear, End를 사용할 수 없습니다. 만일 Buffer 값이 false인데도 불구하고 Flush, Clear, End를 사용하면 다음과 같은 에러 코드를 출력합니다.

```
응답 객체 error 'ASP 0159 : 80004005'
버퍼링 해체
buffer.asp, line 5
버퍼링을 설정해야 합니다.
```

Buffer 값이 true일 때, 즉 ASP 코드를 버퍼에다 모두 저장한 후 한꺼번에 유저의 컴퓨터로 전송하면, 페이지 로딩이 너무 길게 느껴질 수 있습니다. 특히 서버 시스템에 리소스가 부족할 때는 더욱 그렇습니다. 그래서 Flush, Clear, End 메소드를 사용합니다. 먼저 Flush는 호출되는 시점에서 현재까지 버퍼에 저장된 내용을 유저의 컴퓨터에 전송합니다. End는 프로세서를 종료할 때 사용합니다. 마지막으로 Clear는 현재 버퍼에 저장된 내용을 유저에게 전달하지 않고 모두 지워줍니다.

다음 구문은 Buffer의 사용 예를 보여주고 있습니다.

```
<% Response.Buffer = true %>       '버퍼의 저장 시작
....
<html>
<body>
....
....
<% Response.Flush %>      '버퍼에 저장된 내용 전송
...
...
...
<% Response.Flush %>      '버퍼에 저장된 내용 전송
...
...
...
</body>
</html>              '버퍼에 저장된 내용 전송
```

ContentType

ContentType은 웹브라우저에게 현재 파일의 정보 타입이 무엇인지, 어떤 종류의 정보가 전송될 것인지 알려주는 역할을 합니다. 다음은 그 사용 예입니다.

```
<% Response.ContentType="Text/HTML" %>
```

ContentType를 일반적으로 MIME라고 합니다. 윈도우 운영체제에서는 폴더 옵션을 통해 파일의 정보 타입을 확인할 수 있습니다.

Expires, ExpiresAbsolute

Expires는 현재 페이지의 캐시 파일을 지정된 시간이 경과했을 때 갱신하게 해주는 메소드입니다. 즉, 캐시 파일을 삭제하고 갱신된 새로운 웹페이지를 다운 받도록 합니다. 웹브라우저는 캐시에 저장되지 않은 웹페이지를 새롭게 갱신할 페이지로 판단합니다. Expires는 다음 예처럼 사용됩니다.

```
<% Response.Expires=10 %>
```

이 구문에서 10은 시간을 나타내며, 단위는 분(minute)입니다. 만일 페이지 캐싱없이 언제나 새로운 페이지를 받아보고 싶다면 0으로 설정합니다.

또한 지정한 시간에 페이지를 갱신하려면, 다음 예와 같이 ExpiresAbsolute 메소드를 사용합니다. 시간 앞 뒤로 #표시가 붙어 있는 것에 주의하기 바랍니다.

```
<% Response.ExpiresAbsolute=#5/20/2001 00:00:00# %>
```

AddHeader

문자 그대로 HTTP 헤더에 데이터를 추가하는 메소드입니다.

```
<% Response.AddHeader "헤더이름", "헤더 값 "%>
```

Status

웹서버에서 보낸 상태 값을 지정합니다. 웹서버에서 보낸 상태 표시줄을 수정하려면 이 속성을 사용합니다.

```
<% Response.Status = "401 Unauthorized" %>
```

Charset

Charset 속성은 문자의 집합 이름을 응답 객체의 내용 – 형식 헤더에 추가할 때 사용합니다.

```
<% Response.Charset = "euc-kr" %>
```

만일 이 속성을 HTML의 메타(meta) 태그 내에 사용했다면 ASP 코드 내에서 사용할 필요가 없습니다.

Cookies

쿠키는 유저의 클라이언트 컴퓨터에 어떤 정보를 저장해 놓고, 필요할 때마다 꺼내 쓰는 일종의 정보 파일입니다. 'Request 객체'에서 설명했던 Request.Cookies가 쿠키의 정보를 불러오는 것인 데 반해, Response.Cookies는 쿠키의 값을 유저의 컴퓨터에 저장하는 역할을 합니다. 쿠키는 저장되는 위치가 유저의 컴퓨터이기 때문에 서버에는 전혀 무리를 주지 않는 장점이 있습니다.

쿠키에 정보를 저장하기 위한 구문은 다음과 같습니다.

```
<% Response.Cookies("user") = "DDCom" %>
```

이 구문은 user라는 쿠키를 만들고, 그 쿠키에 DDCom이라는 정보를 저장합니다. 이 쿠키 값을 읽어오려면 다음 구문을 사용합니다.

```
<% Request.Cookies("user") %>
```

쿠키는 배열 형식으로도 사용할 수 있는데, 그렇게 되면 더 많은 정보를 저장할 수 있습니다. 쿠키를 배열로 사용하려면 다음 구문처럼 사용합니다.

```
<% Response.Cookies("user")("name") = "DDCom" %>
<% Response.Cookies("user")("age") = "21" %>
<% Response.Cookies("user")("tel") = "025380184" %>
```

이 구문은 user라는 쿠키를 만들고, name, age, tel 항목에 각각의 정보를 저장하게 됩니다. 이렇게 배열 형식으로 사용된 쿠키를 쿠키 사전이라고 합니다. 쿠키 사전은 name, age, tel을 user로 묶을 수 있는 것을 뜻합니다. 예를 들어, 〈%=Request.Cookies("user")%〉 코드로 user 쿠키사전 값을 출력해 보면 tel=025380184&age=21&name=DDCom을 가지고 있음을 알 수 있습니다.

배열 형식으로 저장된 쿠키의 정보를 각각 가져오려면 Request 객체를 사용해 다음 구문처럼 불러올 수 있습니다.

```
<% Request.Cookies("user")("name") %>
<% Request.Cookies("user")("age") %>
<% Request.Cookies("user")("tel") %>
```

다음 예제는 쿠키를 사용한 구문입니다.

예제 2-7
쿠키 사용 예

```
<%
lastdate = Request.Cookies("user")("lastdate")
'이전에 저장 user라는 쿠키의 lastdate 배열을 불러와 lastdate라는 변수에 저장한다.
Response.Cookies("user")("lastdate") = now
'현재 쿠키의  lastdate 배열에 현재 시간을 저장합니다.
Response.Cookies("user")("visitno")  =Request.Cookies("user")("visitno")
+ 1
'현재 쿠키의  visitno 배열에 1을 더합니다.
%>
<HTML>
<HEAD>
```

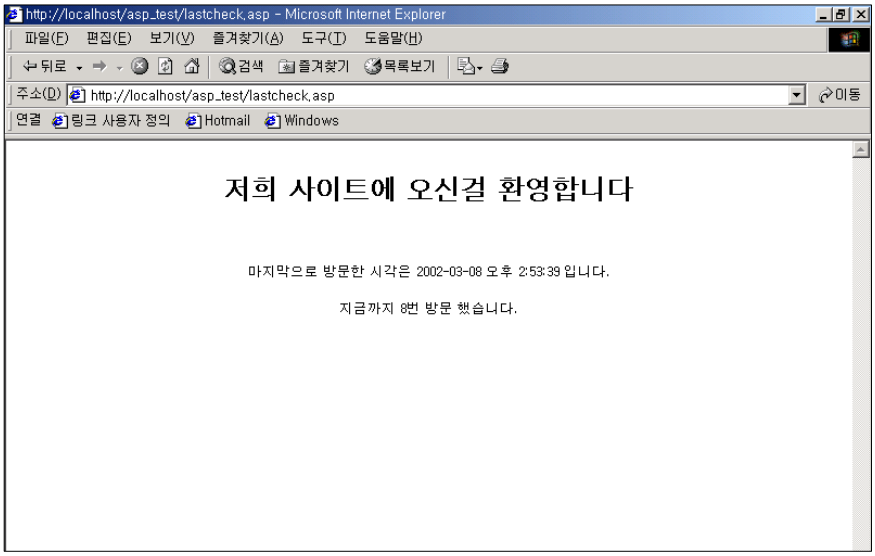
```
</HEAD>
<BODY><br><center><font face="돋움" size="2">
<h2>저희 사이트에 오신걸 환영합니다</h2>
<P> </P>

<!--변수 lastdate를 HTML 문서에 표시합니다.//-->
마지막으로 방문한 시각은 <%=lastdate%> 입니다.<p>

<!--쿠키의 visitno 배열의 값을 HTML 문서에 표시합니다.//-->
지금까지 <%=Request.Cookies("user")("visitno")%>번 방문 했습니다.
</font></center>
</BODY>
</HTML>
```

[예제 2-7]의 결과는 다음 그림과 같습니다. 처음 이 프로그램을 실행시키면 마지막으로 방문한 시간이 표시되지 않을 것입니다. 이에 대한 사항은 [예제 2-8]에서 자세히 다루도록 하겠습니다. 일단 웹브라우저의 새로 고침 아이콘을 여러 번 눌러 방문시간과 방문 횟수의 변화를 살펴보도록 합니다.

그림 2-7
[예제 2-7] 결과 화면



[예제 2-7]의 코드는 초기 접속했을 때 마지막 방문 시간이 출력되지 않는 문제가 있었습니다. 다음의 [예제 2-8]처럼 일부 코드를 수정합니다. 조금 복잡해 보이지만, 쿠키에 대해 완전히 이해하기 위해 어쩔 수 없이 만든 코드입니다.

예제 2-8
쿠키 사용 예(수정)

```
<%
Response.Cookies("user").expires = #12/30/2002 00:00:00#
if Request.Cookies("user")<>"" then
  lastdate = Request.Cookies("user")("lastdate")
  Response.Cookies("user")("lastdate") = now
Response.Cookies("user")("visitno")=Request.Cookies("user")("visitno") + 1
else
  lastdate = "~~~ 어서오세요 오늘이 첫 방문"
  Response.Cookies("user")("lastdate") = now
  Response.Cookies("user")("visitno") = 1
end if
%>
<HTML>
<HEAD>
</HEAD>
<BODY><br><center><font face="돋움" size="2">
<h2>저희 사이트에 오신걸 환영합니다</h2>
마지막으로 방문한 시각은 <%=lastdate%> 입니다.<p>
지금까지 <%=Request.Cookies("user")("visitno")%>번 방문 했습니다.
</font></center>
</BODY>
</HTML>
```

Response.Cookies("user").expires = #12/30/2002 00:00:00#는 지정된 시간에 쿠키의 내용을 만료시킵니다. 즉, 2002년 12월 30일이 되면 현재의 쿠키는 더 이상 사용할 수 없게 되는 것입니다.

if Request.Cookies("user")〈〉"" then은 조건문의 시작으로, "user라는 쿠키에 무엇인가 저장되어 있다면"이라는 뜻입니다. "" 는 아무것도 없는 빈 문자열이고, 〈〉은 not을 의미합니다. 따라서 "비어있지 않다"라는 식으로 해석할 수 있겠죠. 정리하면, if 조건이 참이면 then 이하의 구문을 수행하고, 거짓(쿠키 값이 있을 경우)이면 else 구문 이하를 실행합니다. 참고로 if 조건문의 끝은 End if입니다.

lastdate = Request.Cookies("user")("lastdate")는 lastedate라는 변수를 만들고, 그 변수에 user 쿠키의 lastdate 배열 값을 가져와 저장합니다. 즉, 이 구문은 동일한 유저가 두 번째 접속할 때 실행됩니다. 첫 번째 접속할 때에는 당연히 쿠키 값이 없을 테니까요.

Response.Cookies("user")("lastdate") = now는 user 쿠키의 lastdate 배열에 현재 시간을 저장합니다. 여기서 now는 ASP의 내장 함수로, 시스템의 현재 시간을 가져와 사용합니다.

Response.Cookies("user")("visitno")=Request.Cookies("user")("visitno") + 1은 user 쿠키의 visitno 배열 값을 가져온 다음, 그 값을 visitno 배열에 다시 저장합니다. 여기서 알아둬야 할 것은 ASP 연산식의 순서입니다. 일반적으로 수학에서는 1 + 1 = 2라는 수서로 식이 이루어집니다. 그러나 ASP에서는 2 = 1 + 1로 생각해야 합니다. 항상 등호를 중심으로 왼쪽 항이 '결과', 오른쪽 항이 '연산'이 됩니다.

else는 If 조건식이 거짓일 때 실행되는 구문입니다. If 조건식이 거짓이라는 것은 유저가 처음 방문했다는 것을 의미하며, 따라서 유저의 컴퓨터에 저장된 쿠키가 없다는 것을 의미합니다. 그래서 If 조건식이 거짓이면 else 이하의 구문을 수행하게 됩니다.

lastdate = "~~~ 어서오세요 오늘이 첫 방문"은 lastdate 변수에 문자열을 저장합니다. 이 변수는 첫 방문일 경우에만 표시되는 문구입니다.

Response.Cookies("user")("lastdate") = now는 user 쿠키의 lastdate 배열에 현재 시간을 저장합니다. 그래야 두 번째 방문할 때 처음 방문한 유저가 아님을 알 수 있습니다.

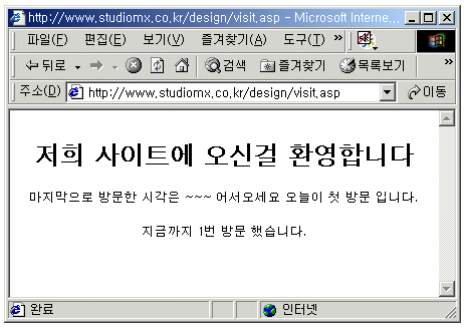
Response.Cookies("user")("visitno") = 1은 user 쿠키의 visitno 배열에 1을 저장합니다. 이 값은 나중에 웹페이지에 표시될 방문횟수이며, 두 번째 방문시에는 이 값을 읽어 들인 후 1을 더해 방문 횟수를 높이게 됩니다.

end if는 조건문을 종료하는 구문입니다. 반드시 필요하니 빠뜨리는 일이 없도록 합니다.

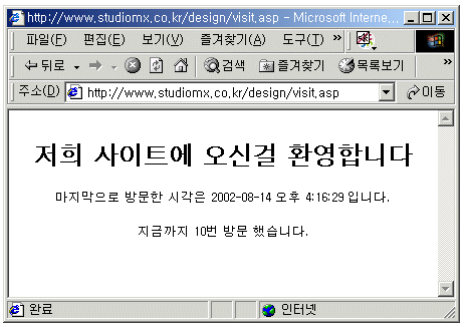
HTML 코드 내에는 두 개의 ASP 코드가 〈%~%〉로 삽입되어 있습니다. 앞에서도 언급한 것처럼, 이 코드는 웹서버에서 실행되어 텍스트로 반환됩니다. 먼저 〈%=lastdate%〉는 HTML이 실행되기 전에 변수로 선언된 lastdate를 출력(=)하는 것입니다. lastdate의 값은 유저가 처음 접속했을 때와 두 번째 이상 접속했을 때를 If 조건문에 의해 따로 설정됩니다. 〈%=Request.Cookies("user")("visitno")%〉 코드는 user 쿠키에서 visitno 배열에 저장된 값을 출력하는 코드로, visitno의 값은 웹페이지가 실행되면서 저장되는 값입니다.

[예제 2-8]을 asp 파일로 저장하여 웹에서 실행하면, 다음 그림처럼 처음 접속했을 때와 두 번 이상 접속했을 때의 화면이 다르게 나타납니다.

그림 2-8
[예제 2-8] 결과 화면



처음 접속했을 때



두 번 이상 접속했을 때

Redirect

Redirect는 다른 웹페이지로 이동할 때 유용하게 사용되는 메소드입니다. HTML에서 사용하는 하이퍼링크라고 생각하면 됩니다. 사용법은 다음과 같습니다.

```
<% Response.Redirect "http://www.ddcom.co.kr" %>
```

Response.Redirect는 다음 예처럼 get 방식으로 정보를 함께 전송할 수도 있습니다.

```
<% Response.Redirect "http://www.ddcom.co.kr/request.asp?id=ddcom" %>
```

Response.Redirect에는 한 가지 단점이 있습니다. 그것은 문서에 프레임이 사용되었을 경우, 프레임을 지정해 줄 수 없다는 점입니다. 그래서 ASP로 제작된 웹페이지는 일반적으로 프레임 구성을 하지 않습니다. 독자가 방문했던 대부분의 웹사이트를 떠올려보면, 프레임을 사용하지 않았다는 것을 기억할 수 있을 것입니다.

그렇다고 프레임으로 구성된 웹페이지를 아예 사용할 수 없는 것은 아닙니다. 프레임으로 구성된 ASP 문서에서 하이퍼링크를 시키려면 자바스크립트를 사용합니다. 다음 예는 ASP에서 하이퍼링크를 설정해주는 코드를 보여주고 있습니다.

```
<Script Language = "javascript">
parent.main_frame.location="left_frme.asp"
parent.top_frame.location="top_frame.asp"
</Script>
```

03 Server 객체

Server 객체는 ASP에서 제공하는 내장 객체 중에서 최상위 객체입니다. 이 객체는 서버의 역할에 관한 사항을 정의할 때 사용되는 것으로, ASP 내장 객체 외의 객체를 생성하거나, 컴포넌트를 이용한 객체, 즉 인스턴스를 생성할 때 주로 사용합니다.

컴포넌트

컴포넌트는 ASP 프로그램밍에 도움을 주는 미리 만들어 놓은 프로그램으로, 개발자는 이 컴포넌트의 인스턴스를 생성하여 어려운 프로그래밍을 하지 않고 속성 지정만으로 프로그램을 끝낼 수 있습니다. 이 책에서도 나중에 업로드 컴포넌트를 사용한 파일 업로드를 배우게 되니 지금은 너무 어려워하지 말고 넘어가기 바랍니다.

Serer 객체의 프라퍼티와 메소드는 다음과 같습니다.

프라퍼티

ScriptTimeout	ASP 프로그램의 오류 발생에 대비하여, 웹 서버가 ASP 스크립트 처리를 위해 소모하는 시간을 제한함

메소드

CreateObject	ASP에서 제공하지 않는 객체나 서버 컴포넌트의 인스턴트 생성
Execute	ASP 페이지 내에서 다른 ASP 페이지에 접근하게 함
HTMLEncode	지정된 문자열이 HTML 코드 형태로 화면에 보이도록 처리함
MapPath	파일이 저장되어 있는 물리적 경로를 알아냄
URLEncode	Escape 문자를 포함한 문자열에 URL을 인코딩해 다른 ASP 페이지로 전달되도록 함

ScriptTimeout

이 프라퍼티는 ASP 프로그램의 실행 시간이 너무 길어 질 경우를 대비해 준비된 것입니다. 예를 들어, ASP 프로그램을 만들다가 개발자의 실수로 무한 루프에 빠졌다면, 어쩔 수 없이 컴퓨터의 전원을 꺼야 합니다. 이 때 ScriptTimeout으로 시간을 지정해 두면 지정된 시간에 프로그램이 강제 종료되고 컴퓨터는 다시 살아납니다. 다음 예는 ScriptTimeout을 사용한 구문입니다.

```
<% Server.ScriptTimeout = 120 %>
```

120은 프로그램이 실행된 뒤, 강제로 멈추게 될 시간 제한으로, 단위는 초(second)입니다.

CreateObject

CreateObject는 서버 객체의 메소드 중에서 가장 많이 사용하는 메소드입니다. ASP에서는 유용하게 사용할 수 있는 프로그램을 컴포넌트로 만들어 사용하기도 하는데, 이 컴포넌트를 사용하기 위해서 CreateObject 메소드를 사용합니다. 예를 들어, Server.CreateObject는 서버에 설치된 컴포넌트를 사용한다는 뜻이 됩니다.

이 책에서도 게시판을 만들거나 메일을 발송하는 작업을 할 때 이 컴포넌트를 사용하게 됩니다. 그 때 더 자세히 다루게 되니 지금은 이해만 하고 넘어가도록 합니다.

다음 예문은 데이터베이스에 접속하기 위해 ADO 컴포넌트를 사용한 것입니다. ADO의 데이터베이스 연결 객체를 사용하기 위해 Server.CreateObject(ADODB.Connection")을 사용하고, 레코드셋 객체를 사용하기 위해 Server.CreatObject("ADODB.RecordSet")을 사용하였습니다.

```
<%
Set DB = Server.CreateObject(ADODB.Connection")
DB.open ("DSN=data;uid=sa;pwd=;")
Set RS = Server.CreatObject("ADODB.RecordSet")
  .
  .
  .
 %>
```

HTMLEncode

HTMLEncode는 단어의 뜻을 풀어보면 알 수 있듯이, HTML 페이지에서 HTML 코드 자체를 표현하기 위해 사용합니다. 예를 들어, 유저의 브라우저에 〈HTML〉이라는 문자를 표시하려면, 실제 코딩은 <HTML>라고 해야 합니다. 다시 말해, 태그 문자나 특수 문자를 웹페이지에 표시하기 위해서는 HTML 문서 내에 특수 코드로 코딩되어야 하는 것입니다. 이것은 ASP에서도 마찬가지입니다. 이것은 앞에서도 설명했듯이, ASP 파일의 결과물은 HTML이기 때문입니다. 따라서 ASP에서도 HTML 태그를 표현하려면 특수 문자로 바꿔줘야 HTML 페이지에 제대로 출력됩니다. [예제 2-9]는 HTMLEncode를 사용한 구문입니다.

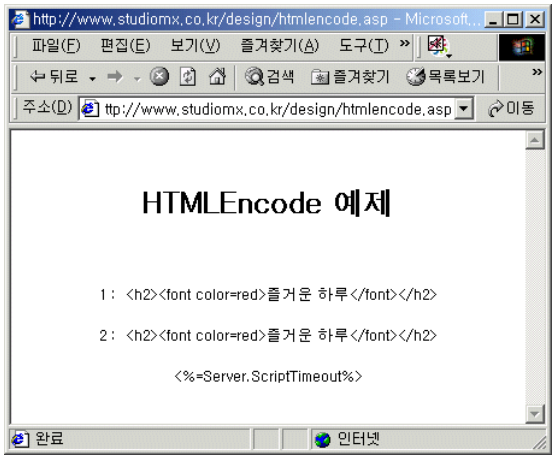
예제 2-9
HTMLEncode를 사용한 구문 예

```
1  <HTML>
2  <BODY>
3  <font face="돋움" size="2">
4  <P> </P>
5  <center>
6  <h2>HTMLEncode 예제</h2>
7  <p>1 :  &lt;h2&gt;&lt;font
   color=red&gt;즐거운 하루&lt;/font&gt;&lt;/h2&gt;</p>
8  <p>2 : 
   <%=Server.HTMLEncode("<h2><font color=red>즐거운 하루</font></h2>")%></p>
9  <p><%=Server.HTMLEncode("<%=Server.ScriptTimeout%\>")%></p>
10  </center></font>
11  </BODY>
12  </HTML>
```

[예제 2-9]의 7행에 〈p〉~〈/p〉 사이에 들어 있는 부분이 HTML 코드를 웹브라우저 화면에 표시하기 위해 사용한 방법입니다. 그리고 8행은 Server.HTMLEncode를 사용하여 ASP 코드에서 HTML 태그를 HTML 문서 내에 표시할 때 사용하는 방법입니다. 마지막으로 9행은 ASP에서 %〉를 ASP 코드로 인식하지 못하게 하기 위해 끝나는 %〉 앞에 \를 붙인 것입니다.

[예제 2-9]를 서버에 업로드하여 웹브라우저에서 읽어보면 다음 그림과 같은 결과를 얻을 수 있습니다. 그림을 보면, HTML 코드와 ASP 코드가 웹브라우저에서 그대로 나타나는 것을 볼 수 있습니다.

그림 2-9
[예제 2-9] 결과 화면



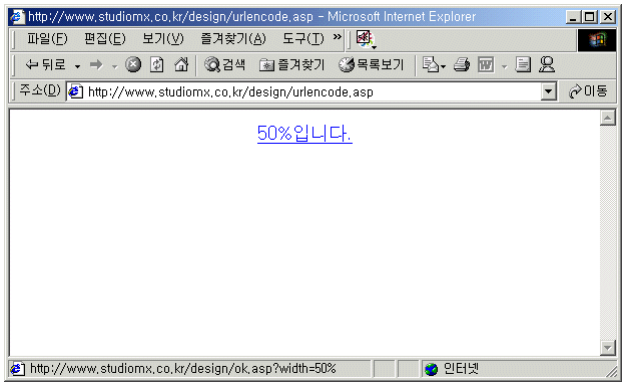
URLEncode

URLEncode는 하이퍼링크로 % 등의 문자를 전송할 때 사용됩니다. 한글 정보를 get 방식으로 전송하면, 해당 한글이 ANSI 코드로 바뀌는데, 이 코드가 바로 % 문자입니다. 따라서 URL에 포함된 % 문자를 인식시켜야 하는데, 이를 위해 URLEncode 메소드를 사용합니다. 다음은 URLEncode 메소드를 사용하고 있는 예입니다.

```
<a href="ok.asp?width=<%=Server.URLEncode("50%")%>">50%입니다.</a>
```

이 코드를 실행하면 그림과 같은 결과를 얻을 수 있습니다. '50%입니다'라는 하이퍼링크 텍스트에 마우스를 위치시켜 봅니다. 그러면 상태표시줄에 표시된 하이퍼링크 주소가 ok.asp?width=50%라는 것을 확인할 수 있습니다.

그림 2-10
URLEncode를 적용한 화면



MapPath

MapPath 메소드는 논리적인 경로를 서버의 물리적인 경로로 바꾸어 주는 역할을 합니다. 여기서 논리적인 경로는 URL 상의 경로를 의미하고, 물리적인 경로는 하드디스크상의 경로를 의미합니다. 예를 들어, 웹페이지 파일은 URL 경로 http://www.studiomx.co.kr/design/index.html로 찾을 수 있습니다. 여기서 index.html 파일은 실제로는 웹서버의 하드디스크, 예를 들면 D:\DDCom_Family\studiomx\design\index.html이라는 경로에 있는 것입니다.

결국 MapPath 메소드는 URL의 경로를 실제 서버의 하드디스크 경로로 바꿔준다는 뜻입니다. 주로 파일을 업로드할 때 사용되는데, 이것은 웹서버의 하드디스크의 경로를

알아야 저장할 수 있기 때문입니다. [예제 2-10]은 MapPath 메소드의 사용법을 보여주고 있습니다.

예제 2-10
MapPath 사용 예

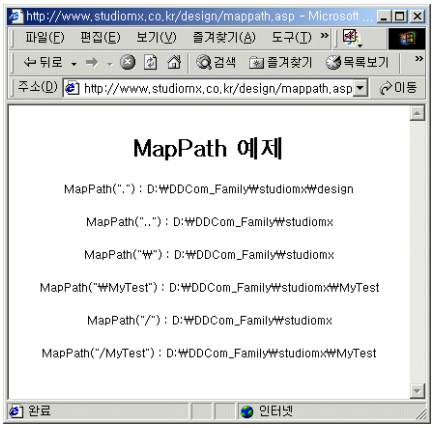
```
1  <HTML>
2  <HEAD></HEAD>
3  <BODY><br><center><font face="돋움" size="2">
4  <h2>MapPath 예제</h2><p>
5  MapPath(".") : <%=Server.MapPath(".")%><p>
6  MapPath("..") : <%=Server.MapPath("..")%><p>
7  MapPath("\") : <%=Server.MapPath("\")%><p>
8  MapPath("\MyTest") : <%=Server.MapPath("\MyTest")%><p>
9  MapPath("/") : <%=Server.MapPath("/")%><p>
10 MapPath("/MyTest") : <%=Server.MapPath("/MyTest")%><p>
11 </font></center>
12 </BODY>
13 </HTML>
```

사이트 경로

기본 웹 사이트 아래 가상 디렉토리를 만들어 작업하는 경우, 기본 웹 사이트의 경로(C:\inetpub\ wwwroot)가 홈 디렉토리 경로가 됩니다.

5행은 현재 웹브라우저에 표시된 ASP 페이지가 있는 디렉토리를 반환합니다. 6행은 현재 ASP 페이지가 있는 위치의 상위 디렉토리를 반환합니다. 7행은 홈 디렉토리의 물리적인 경로, 즉 하드디스크 경로를 반환합니다. 8행의 홈 디렉토리의 하위인 MyTest 디렉토리를 반환합니다. 9행은 7행처럼 홈 디렉토리의 물리적인 경로를 반환합니다. 여기서는 \ 대신에 /를 사용하고 있는데, mappath는 /와 \를 구별하지 않고 동일하게 취급된다는 것을 알 수 있습니다. 마지막으로 10번라인은 홈 디렉토리의 하위인 MyTest 디렉토리를 반환합니다. 다음 그림은 이 소스를 실행했을 때의 결과입니다.

그림 2-11
[예제 2-10] 실행 결과



04 Application 객체

Application 객체는 상당히 개념적인 ASP 내장 객체로, 하나의 웹사이트를 일종의 애플리케이션이라고 보고 이를 통째로 제어하는 객체입니다.

웹사이트를 애플리케이션으로 취급하는 이유는 무엇일까요? ASP는 일반 HTML과 달리 서버에서 실행되어 그 결과물을 클라이언트의 웹브라우저에게 전달합니다. 따라서 실행(execute)되는 하나의 프로그램이기 때문에 애플리케이션이라고 보는 것입니다.

Application 객체가 다른 객체와 다른 점은 이벤트(event)가 존재한다는 것입니다. 이벤트는 애플리케이션이 발생되는 시점을 말합니다. 자바스크립트에서도 이벤트를 사용하죠. 그것과 똑같다고 생각하면 됩니다. Application 객체에서 사용되는 메소드와 이벤트는 다음과 같습니다.

메소드

Lock	Application 변수에 락을 걸어 다른 사용자가 수정할 없도록 제어함
UnlLck	Application 변수의 락을 해제하여 다른 사용자의 수정을 허용

이벤트

OnStart	Application이 시작할 때 발생하는 이벤트 지정
OnEnd	Application이 끝날 때 발생하는 이벤트 지정

Application 객체는 웹사이트 전체를 제어하는 global.asa 파일에서 주로 사용합니다. 이 파일은 웹사이트의 루트에 존재하지만, ASP로 웹사이트를 구축하는 데 반드시 필요한 파일은 아닙니다. 우선 global.asa 파일의 기본 모양을 보도록 하겠습니다(예제 2-11).

예제 2-11
global.asa 기본형

```
1 <SCRIPT LANGUAGE="VBScript" RUNAT="Server">
2 Sub Application_OnStart
3 End Sub
4 Sub Application_OnEnd
```

```
5  End sub
6  Sub Session_OnStart
7  End Sub
8  Sub Session_OnEnd
9  End Sub
10 </SCRIPT>
```

1행의 〈SCRIPT LANGUAGE="VBScript" RUNAT="Server"〉에서 RUNAT= "Server"는 global.asa 파일이 서버에서 실행되는 스크립트 파일이라는 것을 나타내는 것입니다. ASP 코드의 〈% ~ %〉와 비슷한 의미라고 볼 수 있습니다.

2행의 Sub Application_OnStart는 애플리케이션이 시작할 때의 이벤트를 의미합니다. 이 라인 다음부터 애플리케이션이 시작될 때 해야 할 일을 코딩합니다. 3행의 End Sub는 Sub Application_OnStart의 종료 구문으로, 반드시 넣어 주어야 합니다.

4행의 Sub Application_OnEnd는 애플리케이션이 끝날 때의 이벤트입니다. 이 라인 다음부터 애플리케이션이 끝날 때 해야 할 일을 코딩합니다. 5행의 End Sub는 Sub Application_OnEnd의 종료 구문입니다. 반드시 넣어 주어야 합니다.

6행의 Sub Session_OnStart는 '05 Session 객체'에서 다루게 될 Session 객체입니다. Session 객체는 Application 객체와 다르게 유저가 웹사이트에 접속하는 순간 생성되는 객체입니다. 자세한 것은 다음 절에서 알아보기로 하고, Sub Session_OnStart~ End Sub는 사용자마다 각 세션이 시작될 때 해야 할 일을 코딩하는 부분이라는 것만 알아둡니다.

마지막으로 8행의 Sub Session_OnEnd는 사용자마다 각 세션이 끝날 때 해야 할 일을 코딩합니다.

전역 변수의 설정

Application 객체는 웹사이트 전체를 제어하기 때문에 전역 변수 설정이 가능합니다. 전역 변수는 일반 변수와 달리, 한 번 값을 설정하면 웹사이트 전체, 즉 웹사이트 내의 모든 ASP 파일에서 사용할 수 있는 변수입니다. [예제 2-12]는 visit_num이라는 변수를 전역 변수로 선언한 예입니다.

예제 2-12
visit_num가 전역 변수로 선언된 예

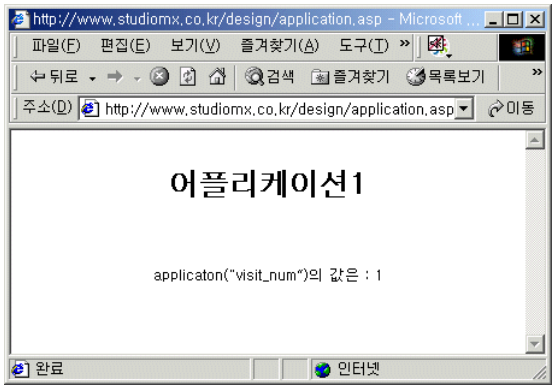
```
1  <% application("visit_num") = 1%>
2  <HTML>
3  <HEAD>
4  </HEAD>
5  <BODY>
6  <center><font face="돋움" size="2">
7  <h2>애플리케이션1</h2>
8  <P> </P>
9  applicaton("visit_num")의 값은 : <%=application("visit_num")%>
10  </font></center>
11  </BODY>
12  </HTML>
```

1행에서는 웹사이트 전체에서 사용할 전역 변수 visit_num을 1로 선언합니다. 그러나 일반적으로 ASP에서 Application 객체의 변수 선언은 예제에서처럼 사용하지 않습니다. 왜냐하면, 전역 변수이므로 애플리케이션이 실행될 때 선언되어야 하고, 해당 파일마다 전역 변수를 선언하면 나중에 관리할 때 해당 파일을 모두 손봐야 어려움이 있기 때문입니다. 따라서 Application 객체의 변수 선언은 global.asa 파일에서 해주는 것이 좋습니다. 여기서 일반 파일 내에서 사용한 것은 Application 객체의 기능을 알아보기 위한 것입니다.

이렇게 선언된 전역변수는 9행에서 〈%=application("visit_num")%〉 코드로 출력되고 있습니다.

위 코드를 실행하면 다음과 같은 결과가 나타납니다.

그림 2-12
[예제 2-12] 실행 결과



Lock과 Unlock

Application 객체는 전역 변수를 생성합니다. 따라서 전역 변수를 사용하고 그 결과 값을 전체 사이트에서 공유합니다. 만일 여러 명의 유저가 웹사이트 내의 한 프로그램을 동시에 실행시켜 결과 값이 동시에 발생한다면 어떻게 될까요? 이 때는 100만분의 1초라도 먼저 만들어진 결과 값이 저장될 것입니다. 그러나 우연치 않게 시간이 완전히 같다면, 심각한 오류를 초래할 수도 있습니다. 이럴 때 사용하는 메소드가 Lock과 Unlock입니다. 앞에서 만든 [예제 2-12]를 조금 변형시켜 Lock과 Unlock의 기능을 알아보겠습니다.

예제 2-13
Lock과 Unlock 사용 예

```
1  <%
2    Application.Lock
3    application("visit_num") =  application("visit_num") + 1
4    Application.UnLock
5  %>
6  <HTML>
7  <HEAD>
8  </HEAD>
9  <BODY>
10  <center><font face="돋움" size="2">
11  <h2>애플리케이션 증가</h2>
13  <P> </P>
14  applicaton("visit_num")의 값은 : <%=application("visit_num")%>
15  </font></center>
16  </BODY>
17  </HTML>
```

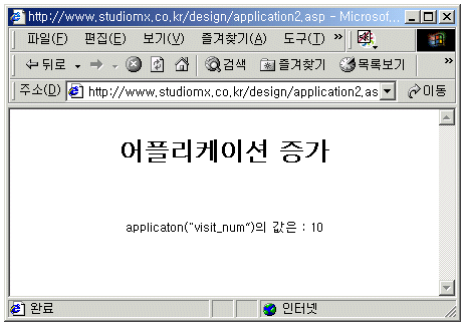
[예제 2-13]은 Application.Lock에 먼저 접근한 유저가 visit_num의 값을 1만큼 증가시키고, Application.UnLock을 수행하기 전까지 다른 유저들은 잠시 기다리게 합니다. 이 시간은 매우 짧기 때문에 유저들이 체감하기 어렵습니다.

2행에서 애플리케이션을 사용하지 못하도록 잠시 잠그게 됩니다. 여기서 락(lock)을 거는 이유는 3행에서 Application 객체로 전역 변수를 선언하는데, 1을 더해서 값을 변경하고 있기 때문입니다. 4행에서는 락을 풀어줍니다. Application에서 선언된 전역 변수 설정이 끝났기 때문에 더 이상 락을 걸어 줄 필요가 없기 때문입니다. 그리고 14행에서는 Application 객체에서 선언된 visit_no의 전역 변수 값을 출력합니다.

[예제 2-13]을 실행해보면 [그림 2-13]처럼 나타납니다. 여기서 [새로 고침] 버튼을 여

러 번 클릭하면 전역 변수 값이 1씩 증가하는 것을 알 수 있습니다.

그림 2-13
[예제 2-13] 실행 결과



05 Session 객체

Application 객체가 웹사이트 전체를 제어하는 데 반해, Session 객체는 웹사이트에 접속한 각 유저들을 제어합니다. Session 객체는 웹사이트에 접속한 각 유저에게 고유한 값을 할당하는데, 다른 유저가 제어할 수 없는 개인적인 영역입니다.

Session은 global.asa 파일에서 사용되며, Session_OnStart를 거쳐 실행되고, Session_OnEnd를 거쳐 소멸됩니다. Session의 주요 프라퍼티, 메소드, 이벤트는 다음과 같습니다.

프라퍼티

SessionID	유저를 식별하기 위해 할당되는 고유의 값.
TimeOut	사용자의 세션이 유지되는 시간을 분 단위로 지정

메소드

Abandon	사용자의 세션을 소멸시킴

이벤트

OnStart	사용자가 처음 방문해 새로운 세션이 발생할 때의 이벤트 지정
OnEnd	사용자의 세션이 끝날 때 발생하는 이벤트 지정

Session을 이해하기 위해 [예제 2-14]와 같은 구문을 만들어 봅니다.

예제 2-14
Session 사용 예

```
<%
  Application.Lock
  Application("visit_num") = Application("visit_num") + 1
  Application.UnLock
  session("visit_num") = session("visit_num") +1
%>
<HTML>
<HEAD>
</HEAD>
```

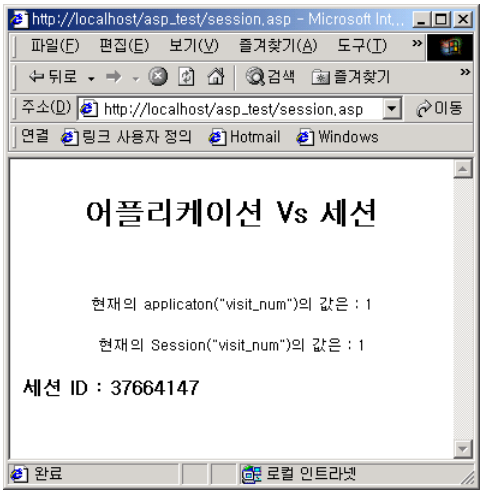
```
10  <BODY>
11  <center><font face="돋움" size="2">
12  <h2>애플리케이션 Vs 세션</h2>
13  <P> </P>
14  현재의 applicaton("visit_num")의 값은 : <%=application("visit_num")%><p>
15  현재의 Session("visit_num")의 값은 : <%=Session("visit_num")%><p>
16  </font></center>
17  <b>세션 ID : <%=session.SessionID%></b>
18  </BODY>
19  </HTML>
```

2행은 Application에 락을 겁니다. 따라서 이후 라인 어디에선가 Application 전역 변수가 변경된다는 것을 알 수 있습니다. 3행에서 Application 전역 변수인 Visit_num 변수에 1을 더해 새로운 Application 전역 변수 visit_num을 선언합니다. 4행에서는 Application에 설정된 락을 풀어줍니다. 그래야 다른 유저가 접속해도 Application 값을 수정할 수 있습니다.

5행은 Session 객체로 visit_num이라는 변수를 선언하고 있는데, 현재 변수에 1을 더하고 있습니다. 단, Application과 Session에서 선언한 변수의 이름이 visit_num이라고 해서 서로 같은 것이 아닙니다. 객체가 다르기 때문에 서로 다른 값으로 사용된다는 점에 주의하기 바랍니다.

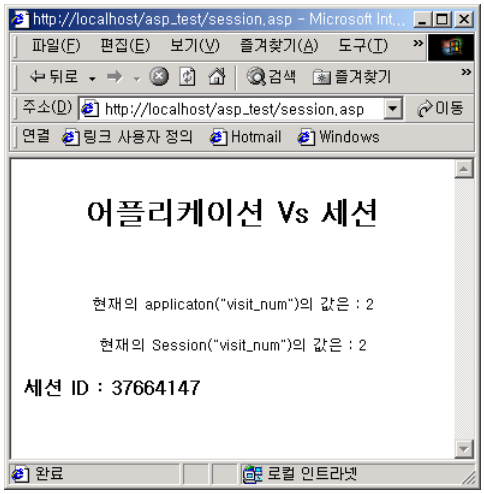
[예제 2-14]의 코드로 되어 있는 웹페이지에 접속하면 다음 그림처럼 나타날 것입니다.

그림 2-14
[예제 2-14] 실행 결과



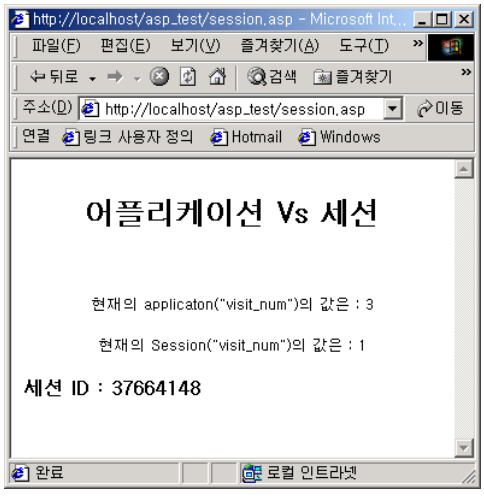
페이지를 새로 고침 해봅니다. 그러면 다음 그림처럼 애플리케이션의 변수 값과 세션의 변수 값이 각각 1씩 증가하는 것을 알 수 있습니다. 여기서 중요한 점은 하단에 표시된 세션 ID의 값은 동일하다는 것입니다. 이를 통해 서버가 동일한 유저로 판단했다는 것을 알 수 있습니다.

그림 2-15
새로 고침했을 때, 변수값이 각각 1씩 증가한다



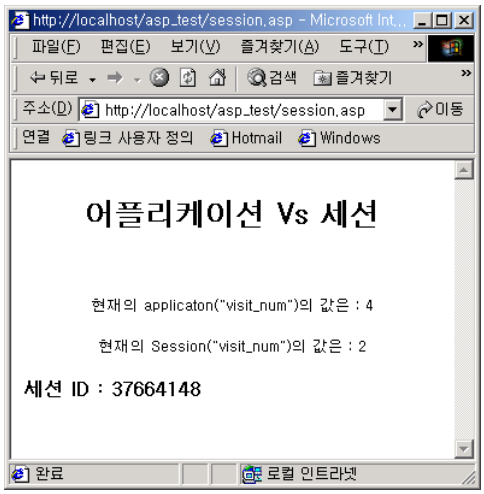
이번에는 현재 열려 있는 웹브라우저를 열어둔 상태에서 다른 웹브라우저를 하나 더 띄워 동일한 페이지에 접속합니다. 그러면 다음 그림처럼 애플리케이션과 세션의 변수 값이 서로 다르게 표시됩니다. 이 때 세션 ID의 값이 바뀐 것을 볼 수 있는데, 서버가 세션이 다른 유저로 판단했기 때문입니다.

그림 2-16
새로운 웹브라우저로 다시 접속했을 때



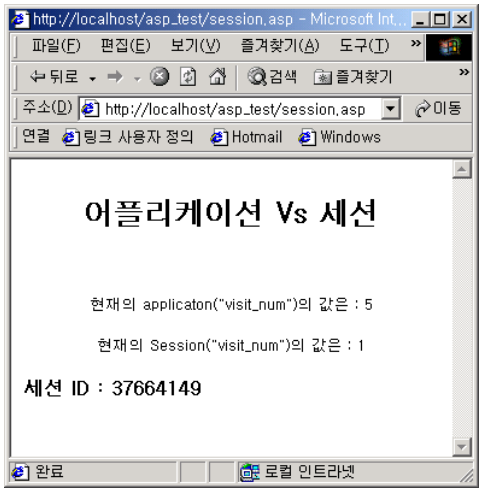
새로 접속한 페이지에서 다시 새로 고침을 해봅니다. 역시 동일한 유저이기 때문에 애플리케이션의 변수와 세션의 변수는 1씩 증가하고, 세션 ID가 변함이 없는 것을 볼 수 있습니다.

그림 2-17
새로운 웹브라우저로 재접속한 후, 다시 새로고침 했을 때. 변수가 1씩 증가한다



그렇다면 이 상태에서 또 다른 웹브라우저를 실행해서 접속하면 어떻게 될까요? 그러면 다음 그림처럼 새로운 세션이 할당될 것입니다.

그림 2-18
세 번째 웹브라우저를 열어 다시 접속했을 때. 세션 변수는 1이 된다.



Session의 역할에 대해 이해하셨습니까? 아직도 이해하지 못한 독자는 계속 실습해보기 바랍니다. 프로그래밍 공부는 실습이 성공의 지름길입니다.

TimeOut

세션에 대한 특별한 정의가 없을 경우, 기본 유지 시간은 20분이며, 유저가 웹사이트에 접속했을 때부터 이 시간이 적용됩니다.

유저가 웹사이트에 접속한 후 20분 동안 아무런 일을 하지 않으면, 세션은 자동으로 소멸됩니다. 또한 유저가 웹사이트에서 나갔을 때는 20분 후에 세션이 자동으로 소멸됩니다. 그렇다면 웹사이트에 나간 후, 20분이 지나지 않았을 때 다시 접속하면 어떻게 될까요? 그 세션은 아직 살아있기 때문에 계속 사용할 수 있습니다.

그렇다면 이 세션 유지 시간을 바꿀 수는 없을까요? 바로 TimeOut 프라퍼티를 사용해서 바꿀 수 있습니다. 다음 예문 처럼 global.asa 파일에서 지정해주면 세션 유지 시간을 변경할 수 있습니다.

```
Session.TimeOut = 10
```

세션과 쿠키의 차이점

세션은 쿠키와 유사한 기능을 가지고 있습니다. 서로 같다고 이해해도 좋습니다. 다만 차이가 있다면, 쿠키는 클라이언트 컴퓨터에 저장되는 반면, Session은 서버 컴퓨터에 저장된다는 것입니다.

Abandon

세션이 유지되는 시간 내에서 세션을 강제 종료할 필요가 있을 때 사용하는 메소드입니다. 주로 웹사이트 로그아웃에 사용됩니다.

Abandon 메소드는 현재 접속한 유저가 갖고 있는 세션 변수 값을 모두 삭제합니다. 그렇다고 해서 접속되어 있는 모든 유저의 세션을 삭제한다는 것은 아닙니다. 오직 abandon 메소드를 실행하는 유저의 세션만 삭제합니다. 참고로 Abandon 메소드가 호출된 페이지는 모든 스크립트 명령이 처리될 때까지 session 값이 삭제되지 않습니다. 즉, Abandon 메소드가 호출된 동일한 페이지 내에서는 session 객체에 저장된 변수를 사용할 수 있습니다. 하지만 이후의 웹 페이지에서는 사용할 수 없습니다. 다음 예문은 세션을 삭제하는 구문입니다.

```
<% Session.Abandon %>
```

global.asa와 Session

globla.asa 파일은 있어도 되고 없어도 된다고 했습니다. 만일 glosal.asa 파일이 없다면, 세션 ID가 어떻게 생성될까요? 세션 ID는 global.asa 파일과는 관계없이 웹사이트에 접속될 때 자동으로 생성됩니다.

세션의 종료 시점에 대해 정리하면 다음과 같습니다.

- TimeOut으로 지정된 시간 동안 아무런 작업을 하지 않았을 경우. 즉, 마지막 작업으로부터 지정된 시간이 경과하면 세션이 소멸됩니다.
- Abandon 명령이 실행된 경우.
- 웹브라우저를 종료하고 지정된 세션 유지 시간을 경과한 경우.
- global.asa 파일을 편집하고 저장했을 경우.
- 웹서버가 중지된 경우.

카운터 만들기

chapter 3

이번 장에서는 1장에서 배운 여러 객체를 활용해 카운터를 만들어 보겠습니다. 특히 Application 객체와 Session 객체의 활용을 중점적으로 다룰 것입니다.

처음에 만드는 카운터는 텍스트로 표시되게 만듭니다. 그런 다음 텍스트형 카운터를 이미지로 바꿔 모양을 내보도록 하겠습니다. 어려울 것 같지만, 생각보다 쉬울 것입니다.

01 Global.asa 파일 만들기

카운터를 만들기 위해 우선적으로 만들어줘야 하는 global.asa 파일을 만듭니다.

카운터는 웹사이트에 접속되는 유저의 수를 기록하는 것입니다. 그런데 일반적인 카운터는 새로 고침을 했을 때 카운트값이 증가하는 문제가 있습니다. 이를 해결하기 위해 ASP에서는 global.asa라는 파일을 사용합니다. 이 파일은 Application, Session 객체를 사용해 유저의 접속을 판단한 다음 카운터를 작성하고, FSO(FileSystemObject) 컴포넌트를 이용하여 카운터 내용을 파일로 저장합니다.

Application_OnStart

Application_OnStart는 웹사이트에 접속을 했을 때, 즉 웹사이트의 애플리케이션이 실행되었을 때 수행되는 구문입니다. 다음의 [예제 3-1]은 FSO 컴포넌트의 인스턴스를 생성하고, 생성된 인스턴스로 카운터가 저장된 파일의 내용을 가져와서 전역 변수에 저장합니다.

예제 3-1
Application_OnStart 사용 예

```
1  Sub Application_OnStart
2     Set FileObject = Server.CreateObject("Scripting.FileSystemObject")
3     VisitCountFileName = "C:\asp_test\visit.txt"
4     Set Out = FileObject.OpenTextFile(VisitCountFileName,1,FALSE,FALSE)
5     Application("count")=Out.ReadLine
6     Application("VisitCountFileName")=VisitCountFileName
7     Application("now_visit") = 0
8  End Sub
```

1행에서 애플리케이션의 시작을 알리는 Application_OnStart 이벤트가 사용되었습니다. 이것은 유저가 웹사이트에 접속했을 때 수행되는 부분입니다.

2행은 FileObject 컴포넌트의 인스턴스를 생성하는데, 인스턴스는 Server 객체를 사용해 FileSystemObject 컴포넌트로 생성됩니다. 인스턴스 생성 방법은 다음과 같습니다.

```
Set 인스턴스 이름 = Server.CreateObject("컴포넌트")
```

3행은 VisitCountFileName이라는 변수를 선언합니다. 변수의 이름이 길지만, 어떤 변수인지 파악하는 데 도움이 될 것입니다. 이 변수에는 카운터 내용이 저장되는 파일의 물리적인 경로가 저장됩니다. 이 경로는 독자가 사용하는 계정이나 웹사이트의 루트로 설정하면 됩니다.

4행은 Out 인스턴스를 생성합니다. 이 인스턴스는 이미 만들어진 FileObject 인스턴스에 OpenTextFile 메소드를 사용하여 VisitCountFileName으로 선언된 변수의 파일을 읽어 들입니다. 여기서 1은 읽기전용으로 읽어오는 것을 의미합니다. 1 대신 2를 사용하면 쓰기 전용으로 읽어 들이고, 8(읽기, 쓰기 가능)을 사용하면 파일을 열어 맨 마지막 부분부터 이어 쓰게 됩니다. 첫 번째 False는 지정된 파일이 없을 때 새 파일을 만들 것인지 지정하는 불린 값입니다. 따라서 새 파일을 만들려면 True를, 그렇지 않으면 False를 사용합니다. 두 번째 False는 파일을 어떤 방식으로 읽어오는지 설정하는 것으로, True면 유니코드(Unicode) 형식으로 파일을 읽어오고, False이면 아스키(ASCII) 코드 형식으로 읽어 옵니다.

5행은 설정된 Out 인스턴스에 ReadLine 메소드를 사용하여 한 줄의 내용을 읽어 들인 다음, 이를 전역 변수 count에 저장합니다. 6행도 전역 변수를 선언하는데, VisitCount FileName으로 설정된 변수를 동일한 이름의 전역변수로 선언합니다. 7행 역시 전역 변수를 선언하는데, now_visit라는 전역변수에 0을 설정합니다. 이 숫자는 카운터의 시작 숫자를 의미합니다.

8행에서 이벤트 핸들러가 종료됩니다.

Session_OnStart

Session_OnStart는 접속한 유저의 세션이 시작될 때, Application_OnStart에서 지정한 전역 변수 값을 각각 1씩 증가시킵니다. 증가된 값은 다시 전역변수로 선언되고, 웹사이트 내의 어디서든지 불러와 사용할 수 있습니다. 그리고 카운터 값은 다시 파일에 저장됩니다. Session_OnStart의 사용 예는 [예제 3-2]와 같습니다.

예제 3-2
Session_OnStart 사용 예

```
1  Sub Session_OnStart
2     Application.lock
3     Application("count")= Application("count")+1
4     Application("now_visit") = Application("now_visit") + 1
5     Application.unlock
6     Session.Timeout = 20
7     Set FileObject = Server.CreateObject("Scripting.FileSystemObject")
8     Set Out = FileObject.CreateTextFile(Application("VisitCountFileName"),
      TRUE)
9     Appication.lock
10     Out.WriteLine(Application("count"))
11     Out.Close
12     Application.unlock
13  End Sub
```

1행의 Sub Session_OnStart는 세션이 시작됨을 알리는 이벤트입니다.

2행은 카운터 전역 변수인 Application("count")와 Application("now_visit")를 1만큼 증가시킬 때 유저의 동시 접근을 막기 위해 Application 객체를 사용해 락을 겁니다. 3행은 카운터 변수인 Application("count")의 수를 1 증가시킵니다. 4행도 마찬가지로 전역 변수인 Application("now_visit")의 수를 1 증가시킵니다.

5행은 Application에 걸린 락을 풀어주고, 6행은 세션의 유지 시간을 20분으로 설정합니다.

7행은 FileObject라는 인스턴스를 생성하는데, 이 인스턴스는 Server 객체를 사용하여 FileSystemObject 컴포넌트로 생성됩니다. 실제 파일이 생성되는 지점입니다. 8행은 Out이라는 인스턴스를 생성합니다. 이 인스턴스는 이미 만들어진 FileObject 인스턴스에 CreatTextFile 메소드를 사용하여 VisitCountFileName으로 선언된 변수의 파일을 만듭니다.

9행은 파일에 동시 저장되는 것을 막기 위해 Application에 락을 걸고, 10행은 Application ("count")의 값을 저장합니다. 11행은 생성된 Out 인스턴스를 소멸시키고, 12행은 Application의 락을 풀어 줍니다.

13행에서 이벤트 핸들러를 종료합니다.

Sub Session_OnEnd

Sub Session_OnEnd는 세션이 종료될 때 실행되는 이벤트로, 세션이 종료되면 현재 접속자 수에서 1을 감소시키는 역할을 합니다.

예제 3-3
Sub Session_OnEnd 사용 예

```
1 Sub Session_OnEnd
2    Application.lock
3    Application("now_visit") = Application("now_visit") - 1
4    Application.unlock
5 End Sub
```

1행의 Sub Session_OnEnd는 세션이 끝남을 알리는 이벤트입니다.

2행은 전역 변수인 Application("now_visit")의 값이 동시에 감소되지 않도록 락을 겁니다. 3행에서 Application("now_visit")의 값을 1 감소시키고, 4행에서 Application의 락을 풀어 줍니다.

5행은 이벤트 핸들러를 종료합니다.

지금까지 작성된 소스 파일, 즉 global.asa는 다음과 같습니다.

```
<SCRIPT LANGUAGE="VBScript" RUNAT="Server">
Sub Application_OnStart
    Set FileObject = Server.CreateObject("Scripting.FileSystemObject")
    VisitCountFileName = "C:\asp_test\visit.txt"
    Set Out = FileObject.OpenTextFile(VisitCountFileName,1,FALSE,FALSE)
    Application("count")=Out.ReadLine
    Application("VisitCountFileName")=VisitCountFileName
    Application("now_visit") = 0
End Sub
Sub Application_OnEnd
End sub
Sub Session_OnStart
    Application.lock
    Application("count")= Application("count")+1
    Application("now_visit") = Application("now_visit") + 1
    Application.unlock
```

```
    Session.Timeout = 20
    Set FileObject = Server.CreateObject("Scripting.FileSystemObject")
    Set Out = FileObject.CreateTextFile(Application("VisitCountFileName"),
TRUE)
    Application.lock
    Out.WriteLine(Application("count"))
    Out.Close
    Application.unlock
End Sub
Sub Session_OnEnd
    Application.lock
    Application("now_visit") = Application("now_visit") - 1
    Application.unlock
End Sub
</SCRIPT>
```

02 웹문서에 카운터 삽입하기

global.asa 파일 작성이 끝났습니다. 이제 웹문서에 카운터를 표시하는 작업만 남았습니다.

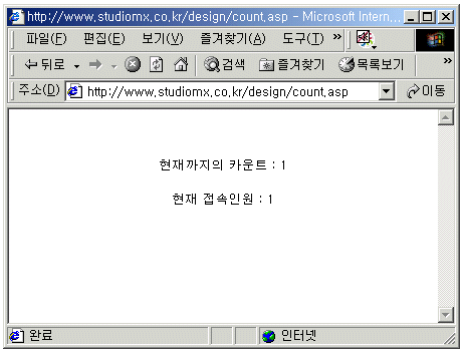
웹문서에 카운터를 표시하려면 [예제 3-4]처럼 만들면 됩니다. 그리고 ASP 구문이 실행되어야 하므로 파일의 확장자는 반드시 ASP로 해야 합니다.

예제 3-4
웹문서에 카운터를 삽입하는 파일

```
<HTML>
<BODY >
<center><font face="돋움" size="2">
<p>  </p>
현재까지의 카운트 : <%=application("count")%> <p>
현재 접속인원 : <%=Application("now_visit")%><p>
</font></center>
</BODY>
</HTML>
```

[예제 3-4]를 실행하면 웹브라우저에 다음 그림처럼 나타납니다. 이 파일을 실행하기 전에global.asa 파일은 사이트의 루트에 업로드시켜 두어야 한다는 것을 잊지 말기 바랍니다.

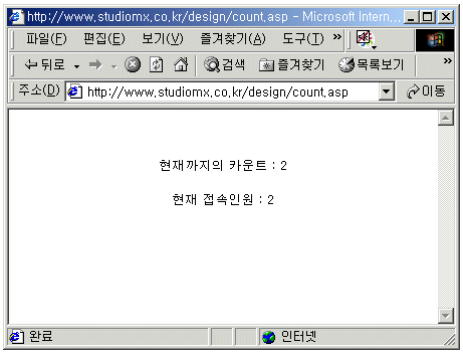
그림 3-1
[예제 3-4] 실행 결과. 카운터가 삽입된 화면



주의할 점은 [예제 3-1]의 3행에서 지정했던, 카운터의 내용이 저장될 텍스트(visit.txt) 파일이 반드시 존재해야 한다는 것입니다.

다음 그림처럼 웹브라우저를 하나 더 실행한 후, 동일한 페이지에 접속해 봅니다. 그러면 웹서버는 다른 세션을 발생시켜 현재 접속 인원을 1 증가시키고, 접속자 수(카운트)도 1 증가시킵니다. 그리고 이 접속자 수는 visit.txt 파일에 저장됩니다.

그림 3-2
새로운 웹브라우저로 다시 접속했을 때



03 이미지 카운터 삽입

앞에서 만든 카운터는 텍스트를 화면에 출력해 주는 것이었습니다. 이 소스를 응용하여 이미지를 카운터로 출력하는 ASP 프로그래밍을 해보겠습니다.

우선 카운터로 사용할 이미지 파일이 존재해야 합니다. 1부터 0까지 총 10개의 이미지를 만들고, 현재 파일이 저장되는 폴더 하위에 images라는 폴더를 만들어 1.jpg ~ 0.jpg라는 이름으로 저장합니다. 이미지들은 독자가 직접 멋있게 만들어서 사용하기 바랍니다.

소스에 사용할 이미지가 만들어졌으면, [예제 3-4]를 다음과 같이 수정합니다.

예제 3-5
이미지 카운터 삽입
(예제 3-4 응용)

```
<%
Count = application("count")
for i = 1 to len(Count)
select case Mid(Count, i, 1)
case "1"
ImgCount = ImgCount & "<img src = 'images/1.jpg' align='middle' >"
case "2"
ImgCount = ImgCount & "<img src = 'images/2.jpg' align='middle' >"
case "3"
 ImgCount = ImgCount & "<img src = 'images/3.jpg' align='middle' >"
 case "4"
 ImgCount = ImgCount & "<img src = 'images/4.jpg' align='middle' >"
 case "5"
 ImgCount = ImgCount & "<img src = 'images/5.jpg' align='middle' >"
 case "6"
 ImgCount = ImgCount & "<img src = 'images/6.jpg' align='middle' >"
 case "7"
 ImgCount = ImgCount & "<img src = 'images/7.jpg' align='middle' >"
 case "8"
 ImgCount = ImgCount & "<img src = 'images/8.jpg' align='middle' >"
 case "9"
 ImgCount = ImgCount & "<img src = 'images/9.jpg' align='middle' >"
 case "0"
 ImgCount = ImgCount & "<img src = 'images/0.jpg' align='middle' >"
 End select
```

```
26  Next
27  %>
28  <HTML>
29  <BODY >
30  <center><font face="돋움" size="2">
31  <p>  </p>
32  현재까지의 카운트 : <%=application("count")%> <p>
33  현재까지의 카운트 : <%=ImgCount%> <p>
34  현재 접속인원 : <%=Application("now_visit")%>
35  </font></center>
36  </BODY>
37  </HTML>
```

2행은 application 객체를 통해 count라는 전역 변수를 선언합니다.

3행은 for~next 반복문으로, i 값이 to 다음에 오는 값이 될 때까지 반복합니다. i는 1부터 시작하는데, to 다음에는 len(Count) 이 있습니다. 여기서 len( ) 함수는 문자의 자릿수를 숫자로 바꿔주는 역할을 합니다. 즉, count 변수에 저장된 접속수의 자릿수만 뽑아내는 역할을 하는 것입니다.

Len( ) 함수

형식: Len(문자열)
예문: Len("Seoul")은 숫자 5로 표시됩니다.

4행은 select case ~ end select 조건문으로, select case 다음에 주어진 값을 5행부터 case에 대입해 비교하게 됩니다. 그런데 select case 코드를 보면, Mid(Count, i, 1)라고 되어 있습니다. 여기서 사용된 Mid( ) 함수는 문자열에서 일부분만 뽑아내는 역할을 합니다.

Mid( ) 함수

형식: Mid(문자열, 시작 위치, 문자 길이)
예문: Mid("seoul", 2, 2)는 eo로 표시됩니다.

Mid 함수의 ( ) 안에 있는 첫 번째 count는 전역변수로 저장된 총 접속수입니다. 예를 들어 접속수가 123이면 count의 값이 123이 됩니다. 다음에 있는 i는 for 구문에서 사용한 i입니다. 즉, count에 저장된 접속수의 자릿수를 의미하는 것입니다. 이 값은 Mid 함수에서 뽑아낼 문자의 시작 자릿수를 의미합니다. 그리고 마지막의 1은 Mid 함수에서 뽑아낼 문자의 끝나는 자릿수를 의미합니다.

예를 들어, 총 접속자 수가 123일 겨우, 처음 3행의 식은 for i=1 to 3이 됩니다. 그리고 for~next 문이 한 번 실행되면 4행은 select case Mid(123, 1,1)가 됩니다. 그러면 case의 값은 1이 됩니다. 따라서 5행의 case "1"이라는 식과 비교했을 때 값이 일치하기 때문에 6행의 ImgCount 변수에 이미지 파일명(1.jpg)이 포함된 〈img〉 태그가 저장됩니다. 이렇게 해서 25행의 End select까지 진행됩니다. 다른 csae는 일치되는 값이 없기 때문에 모두 건너뛰게 됩니다.

26행에서는 아직 for 구문이 끝나지 않았으므로 3행으로 되돌아간 후, 이번에는 i=2라는 식을 수행합니다. 이 작업을 i=3이 될 때까지 반복하게 됩니다.

for 구문이 모두 끝나면 ImgCount에는 세 개의 〈img〉 태그가 저장됩니다. 그리고 33행에서 〈%=ImgCount%〉라는 ASP 코드로 HTML 내에 출력됩니다.

코딩이 끝나면, 확장자를 ASP로 해서 저장한 다음 웹브라우저로 접속해 봅니다. 다음 그림처럼 이미지 카운터가 실행됩니다.

그림 3-3
[예제 3-5] 실행 결과. 이미지 카운터가 삽입된 화면