Apache HTTP Server Version 2.4

아파치 투토리얼: CGI를 사용한 동적 페이지 생성

이 문서는 최신판 번역이 아닙니다. 최근에 변경된 내용은 영어 문서를 참고하세요.

참고

소개

관련된 모듈	관련된 지시어
mod_alias mod_cgi	AddHandler Options ScriptAlias

CGI (Common Gateway Interface)는 웹서버가 보통 CGI 프로그램 혹은 CGI 스크립트라고 부르는, (웹페이지 내용을 만드는) 외부 프로그램과 통신하는 방법을 정의한다. 웹사이트에서 동적인 페이지를 만드는 가장 흔하고 간단한 방법이다. 이 문서는 아파치 웹서버에 CGI를 구성하는 방법을 소개하고, CGI 프로그램을 작성해본다.

CGI를 허용하도록 아파치 설정하기

CGI 프로그램이 올바로 동작하려면 CGI 실행이 가능하도록 아파치를 설정해야 한다. 설정하는 방법은 여러가지다.

ScriptAlias

ScriptAlias 지시어를 사용하면 아파치는 특정 디렉토리를 CGI 프로그램용으로 둔다. 아파치는 이 디렉토리에 있는 모든 파일이 CGI 프로그램이라고 가정하여 클라이언트가 자원을 요청하면 자원을 실행하려고 시도한다.

ScriptAlias 지시어는 다음과 같이 사용한다.

```
ScriptAlias /cgi-bin/ /usr/local/apache2/cgi-bin/
```

위 예제는 아파치를 기본 장소에 설치한 경우 httpd.conf 설정파일에 있는 내용이다. ScriptAlias 지시어는 Alias 지시어와 같이 URL 앞부분을 특정 디렉토리로 대응한다. Alias와 ScriptAlias는 보통 DocumentRoot 디렉토리 밖에 있는 디렉토리에 사용한다. Alias와 ScriptAlias의 차이점은 ScriptAlias가 추가로 URL 앞부분으로 시작하는 모든 파일을 CGI 프로그램으로 취급하는 점이다. 그래서 위의 설정은 아파치에게 /cgi-bin/으로 시작하는 자원을 요청하면 /usr/local/apache2/cgi-bin/ 디렉토리에서 찾아서 CGI 프로그램으로 처리하라고 알린다.

예를 들어, URL `http://www.example.com/cgi-bin/test.pl`을 요청하면 아파치는 `/usr/local/apache2/cgi-bin/test.pl` 파일을 실행하여 결과를 반환한다. 물론 파일이 존재하고 실행 가능하며 어떤 방법으로든 출력을 해야 한다. 그렇지 않으면 아파치는 오류문을 보낸다.

ScriptAlias 디렉토리 밖에 있는 CGI

보통 보안상 이유때문에 CGI 프로그램은 `ScriptAlias`한 디렉토리에 한정한다. 그래서 관리자는 누가 CGI 프로그램을 사용할 수 있는지 엄격히 감독할 수 있다. 그러나 적당한 보안조치를 취했다면 아무 디렉토리에서나 CGI 프로그램을 실행하지 않을 이유가 없다. 예를 들어, `UserDir` 지시어를 사용하여 사용자가 자신의 홈디렉토리에 웹페이지를 가지는 경우를 가정하자. 사용자가 자신의 CGI 프로그램을 사용하고 싶은데 `cgi-bin` 디렉토리에 접근권한이 없다면, 다른 곳에서라도 CGI 프로그램을 실행하고 싶을 것이다.

아무 디렉토리에서나 CGI 실행을 허용하려면 두 과정이 필요하다. 먼저, `AddHandler`나 `SetHandler` 지시어를 사용하여 `cgi-script` 핸들러를 작동해야 한다. 두번째로, `Options` 지시어에 `ExecCGI`를 지정해야 한다.

Options를 사용하여 명시적으로 CGI 실행을 허용하기

서버의 주설정파일에 직접 `Options` 지시어를 사용하여 특정 디렉토리에서 CGI 실행을 허용할 수 있다.

```
<Directory /usr/local/apache2/htdocs/somedir>
   Options +ExecCGI
</Directory>
```

위 지시어로 아파치는 CGI 파일의 실행을 허용한다. 어떤 파일이 CGI 파일인지도 서버에게 알려야 한다. 다음 `AddHandler` 지시어는 서버에게 확장자가 `cgi`나 `pl`인 파일은 모두 CGI 프로그램이라고 알린다.

```
AddHandler cgi-script .cgi .pl
```

.htaccess 파일

`.htaccess` 투토리얼 (↗ htaccess.html) 은 `httpd.conf`에 접근권한이 없는 경우에 CGI 프로그램을 사용할 수 있는 방법을 알려준다.

사용자 디렉토리

아래 설정을 사용하면 사용자 디렉토리에서 `.cgi`로 끝나는 파일을 CGI 프로그램으로 실행한다.

```
<Directory /home/*/public_html>
   Options +ExecCGI
   AddHandler cgi-script .cgi
</Directory>
```

다음을 사용하면 사용자 디렉토리의 `cgi-bin` 하위디렉토리에 있는 모든 파일을 CGI 프로그램으로 인식한다.

```
<Directory /home/*/public_html/cgi-bin>
   Options ExecCGI
   SetHandler cgi-script
</Directory>
```

CGI 프로그램 작성하기

``일반적인" 프로그래밍과 CGI 프로그래밍 사이에는 두가지 주된 차이점이 있다.

첫번째 차이는 CGI 프로그램은 다른 출력을 하기전에 먼저 MIME-type 헤더를 출력해야 한다는 점이다. HTTP 헤더는 클라이언트에게 클라이언트가 어떤 내용을 받게될지 미리 알린다. 보통 다음과 같다.

```
Content-type: text/html
```

두번째 차이는 HTML 혹은 브라우저가 보여줄 수 있는 형식으로 출력해야 한다는 점이다. 대부분의 경우 HTML을 출력하지만, 때때로 gif 그림과 같이 HTML이 아닌 내용을 출력하는 CGI 프로그램을 작성하는 경우도 있다.

두가지를 제외하고는 CGI 프로그램 작성은 이미 만들어 보았을 다른 프로그램들과 매우 비슷하다.

처음으로 만든 **CGI** 프로그램

다음은 브라우저에 한 줄을 찍는 CGI 프로그램 예제다. 그대로 `first.pl`이라는 파일에 저장하고, `cgi-bin` 디렉토리에 복사한다.

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello, World.";
```

Perl에 익숙하지 않더라도 무슨 일이 일어나는지 알 수 있다. 첫번째 줄은 아파치(혹은 사용하는 쉘)에게 `/usr/bin/perl` 위치에 있는 인터프리터을 사용하여 이 프로그램 파일을 실행하라고 알린다. 두번째 줄은 방금 말한 content-type 선언을 출력하고 carriage-return 줄바꿈을 두번 출력한다. 그러면 헤더 뒤에 HTTP 헤더의 끝을 뜻하는 빈줄이 생기고, 본문이 시작한다. 세번째 줄은 "Hello, World." 문자열을 출력한다. 이것으로 끝이다.

브라우저를 실행하고 주소를 입력한다

```
http://www.example.com/cgi-bin/first.pl
```

파일 장소를 입력하면, 브라우저창에 `Hello, World.` 한 줄이 보인다. 흥분되지는 않지만, 한번 동작하는 것을 보았으니 이제 다른 것을 시도해 볼 수 있다.

그러나 아직 동작하지 않아요!

웹에서 CGI 프로그램에 접근할때 브라우저에 나올 수 있는 내용은 기본적으로 네가지다.

CGI 프로그램의 출력
: 좋다! 모든 것이 잘 동작한다는 뜻이다. 출력은 정확하지만 브라우저가 올바로 처리하지 못한다면, CGI 프로그램에서 올바른 `Content-Type`을 설정하였는지 확인한다.

CGI 프로그램 소스코드 혹은 **"POST Method Not Allowed"** 문구
: CGI 프로그램을 실행하도록 아파치를 적절히 설정하지 않았다는 뜻이다. 아파치 설정하기 절을 다시 읽고 빼먹은 부분이 있는지 찾아봐라.

"Forbidden"으로 시작하는 문구
: 권한 문제가 있다는 뜻이다. 아파치 오류 로그와 아래 파일권한 절을 확인하라.

"Internal Server Error"라는 문구
: 아파치 오류 로그를 보면 아마도 CGI 프로그램이 출력한 오류문과 함께 "Premature end of script headers"가 보일 것이다. 이 경우 아래 내용들을 하나씩 확인하여 어떤 이유로 CGI 프로그램이 적절한 HTTP 헤더를 출력하지 못했는지 알아본다.

파일권한

서버는 당신과 동일한 계정으로 동작하지 않음을 명심하라. 즉, 서버가 시작하면 서버는 비특권 사용자 권한(보통 `nobody`나 `www`)으로 동작한다. 그래서 당신이 소유한 파일을 실행하려면 권한이 필요하다. 파일에 `nobody`가 실행하기에 충분한 권한을 주기위해 보통 모두에게 파일의 실행 권한을 준다.

```
chmod a+x first.pl
```

또, 프로그램이 다른 파일을 읽거나 쓴다면 이 파일에도 적절한 권한이 필요하다.

경로 정보와 환경

명령행에서 프로그램을 실행하면 자동으로 어떤 정보가 쉘로 전달된다. 예를 들어, `PATH`는 쉘에게 당신이 말한 파일을 찾을 장소를 알려준다.

웹서버가 프로그램을 CGI 프로그램으로 실행할때는 PATH가 다를 수 있다. (예를 들어, sendmail 같이) CGI 프로그램 안에서 실행하는 명령어는 완전한 경로로 명시해야 쉘이 명령어를 찾을 수 있다.

경로 문제는 다음과 같이 CGI 프로그램 첫번째 줄에 나오는 스크립트 인터프리터 (보통 perl) 경로에서 자주 발생한다.

```
#!/usr/bin/perl
```

실제로 인터프리터의 경로인지 확인한다.

또, CGI 프로그램이 다른 환경변수 (↗ #env) 를 사용한다면 아파치가 이 변수들을 프로그램에게 전달해야 한다.

프로그램 오류

CGI 프로그램이 실패하는 경우 대부분 프로그램 자체 문제때문이다. 특히 위의 두가지 실수를 하지 않았고 이 글을 계속 보고 있다면 더더욱 그렇다. 먼저 웹서버에서 실행하기 전에 명령행에서 프로그램을 실행해본다. 예를 들어, 다음과 같이 실행한다.

```
cd /usr/local/apache2/cgi-bin
./first.pl
```

(perl 인터프리터를 실행하지 마라. 쉘과 아파치는 스크립트 첫번째 줄에 있는 경로 정보 (↗ #pathinformation) 를 사용하여 인터프리터를 찾아야 한다.)

프로그램은 제일 먼저 Content-Type을 포함한 HTTP 헤더들을 출력하고 빈 줄을 출력해야 한다. 다른 것을 출력한다면 웹서버에서 실행할 경우 아파치는 Premature end of script headers를 반환한다. 자세한 내용은 위의 CGI 프로그램 작성하기 (↗ #writing) 를 참고하라.

오류 로그

오류 로그는 당신 편이다. 무언가 잘못되면 오류 로그에 문구가 생긴다. 오류 로그를 제일 먼저 살펴봐야 한다. 웹사이트를 호스팅하는 곳에서 오류 로그를 보지 못하게 한다면, 아마도 다른 업체를 알아봐야 한다. 오류 로그를 보는 방법을 익히면, 대부분의 문제를 빨리 파악하여 해결할 수 있다.

Suexec

suexec (↗ ../suexec.html) 지원 프로그램을 사용하면 어떤 가상호스트 혹은 어떤 사용자 디렉토리에 있는지에 따라 CGI 프로그램을 다른 사용자 권한으로 실행할 수 있다. Suexec는 매우 엄격하게 권한을 검사하며, 검사를 하나라도 통과하지 못하면 CGI 프로그램을 실행하지 않고 Premature end of script headers를 반환한다.

suexec를 사용하고 있는지 알려면 apachectl -V를 실행하여 SUEXEC_BIN 위치를 확인한다. 아파치가 시작할때 그 장소에서 suexec 실행파일을 발견하면, suexec를 사용할 수 있다.

suexec를 완전히 이해하지 못했다면 사용해서는 안된다. suexec를 사용하지 않으려면 SUEXEC_BIN 위치에 있는 suexec 실행파일을 지우고 (혹은 파일명을 바꾸고) 서버를 재시작하면 된다. suexec (↗ ../suexec.html) 에 대해 읽은 다음 그래도 사용하고 싶다면, suexec -V를 실행하여 suexec 로그파일 위치를 알아내고 로그파일에서 당신이 어떤 규칙을 어기고 있는지 찾는다.

뒤에서는 무슨 일이 벌어지는가?

CGI 프로그래밍에 익숙해질수록 뒤에서 벌어지는 일을 이해하면 도움이 된다. 구체적으로 브라우저와 서버가 서로 통신하는 방법을 말하는 것이다. 몰라도 "Hello, World."를 출력하는 프로그램을 작성할 수 있지만 이런 프로그램은 별로 쓸모가 없기때문이다.

환경변수

환경변수는 당신이 컴퓨터를 사용하는 동안 당신 주위를 떠다니는 값이다. 환경변수는 path (컴퓨터가 당신이 입력한 명령어에 해당하는 실제 파일을 찾는 장소), 사용자명, 터미널 종류와 같이 유용한 정보다. 일반적인 환경변수를 모두 보려면 명령행 프롬프트에서 env를 입력한다.

CGI를 실행할때도 서버와 브라우저는 각자의 환경변수를 서로 교환한다. 이 정보에는 브라우저 종류 (Netscape, IE, Lynx), 서버 종류 (아파치, IIS, WebSite), 실행하는 CGI 프로그램명 등이 있다.

CGI 프로그래머는 이런 변수들을 사용할 수 있고, 환경변수는 클라이언트-서버 통신에는 일부분을 차지한다. 전체 필수 변수 목록은 http://hoohoo.ncsa.uiuc.edu/cgi/env.html (↗ http://hoohoo.ncsa.uiuc.edu/cgi/env.html) 에 있다.

아래 간단한 Perl CGI 프로그램은 자신에게 전달된 모든 환경변수를 보여준다. 아파치 배포본의 `cgi-bin` 디렉토리에 이와 비슷한 프로그램이 두개 있다. 몇몇 변수는 필수이고 나머지는 선택적이다. 그래서 공식 목록에 없는 변수도 보인다. 또, 아파치는 기본적으로 제공하는 환경변수 외에 여러가지 방법으로 직접 환경변수를 추가할 수 있다 (↗ ../env.html) .

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
foreach $key (keys %ENV) {
   print "$key --> $ENV{$key}<br>";
}
```

STDIN과 STDOUT

또, 서버와 클라이언트는 표준입력(STDIN)과 표준출력(STDOUT)으로 통신한다. 일상적인 경우 STDIN은 키보드나 프로그램이 처리하는 파일을 나타내고, STDOUT은 보통 콘솔이나 화면을 뜻한다.

CGI 프로그램에게 웹 양식(form)을 POST하면 양식에 입력한 자료를 특별한 형식으로 묶어서 CGI 프로그램의 STDIN으로 전달한다. 그러면 프로그램은 키보드나 파일에서 얻은 자료를 처리하듯이 자료를 처리할 수 있다.

"특별한 형식"은 매우 간단하다. 항목 이름과 값을 등호(=)로 연결하고, 항목 이름과 값의 쌍들을 서로 앤퍼샌드(&)로 연결한다. 공백, 앰퍼샌드, 등호 같은 부자연스러운 문자는 혼동하지 않도록 16진수로 변환한다. 완전한 자료 문자열은 다음과 같이 생겼다.

```
name=Rich%20Bowen&city=Lexington&state=KY&sidekick=Squirrel%20Monkey
```

종종 URL 뒤에서 이런 문자열을 보게 된다. 이 경우 서버는 문자열을 QUERY_STRING이라는 환경변수에 저장한다. 이를 GET 요청이라고 한다. FORM 태그의 METHOD 속성을 지정하여 HTML 양식(form)이 자료를 GET할지 POST할지 결정한다.

이제 프로그램은 이런 문자열을 유용한 정보로 쪼개야 한다. 다행히도 이런 자료 처리를 돕고 CGI 프로그램의 다른 여러 면을 살피는 라이브러리와 모듈들이 있다.

CGI 모듈/라이브러리

CGI 프로그램을 작성할때 지루한 작업을 대신해주는 코드 라이브러리 혹은 모듈을 사용할지 고려해봐야 한다. 이런 것을 사용하면 버그가 줄고 더 빨리 프로그램을 개발할 수 있다.

Perl로 CGI 프로그램을 작성한다면 CPAN (↗ http://www.cpan.org/) 에서 관련 모듈들을 찾을 수 있다. CGI 개발에 가장 널리 사용되는 모듈은 `CGI.pm`이다. 대부분의 프로그램에 충분한 최소 기능을 구현한 `CGI::Lite`도 고려해 볼 수 있다.

C로 CGI 프로그램을 작성한다면 선택의 여지가 많다. 이중 하나가 http://www.boutell.com/cgic/ (↗ http://www.boutell.com/cgic/) 에 있는 CGIC 라이브러리다.

더 많은 정보...

웹에 매우 많은 CGI 정보가 있다. 뉴스그룹 comp.infosystems.www.authoring.cgi (↗ news:comp.infosystems.www.authoring.cgi) 에서 여러 사람들과 CGI 문제를 논의할 수 있다. HTML Writers Guild의 -servers 메일링리스트는 질문에 대한 답을 찾기에 훌륭한 장소다. http://www.hwg.org/lists/hwg-servers/ (↗ http://www.hwg.org/lists/hwg-servers/) 에서 더 많은 것을 알 수 있다.

그리고 물론 CGI 프로그램 동작에 대한 모든 내용을 설명한 CGI 규약을 읽어야 할지도 모른다. NCSA (↗ http://hoohoo.ncsa.uiuc.edu/cgi/interface.html) 에 원본 문서가 있고, 수정한 초안은 Common Gateway Interface RFC 프로젝트 (↗ http://web.golux.com/coar/cgi/) 에 있다.

메일링리스트나 뉴스그룹에 현재 격고 있는 CGI 문제에 대해 질문할때는 발생한 현상과 원래 기대한 결과, 실제로 발생한 현상이 어떻게 다른지, 사용하는 서버, CGI 프로그램을 작성한 언어, 가능하면 해당 코드를 자세히 적어라. 그러면 해결책을 찾기 쉬워진다.

아파치 소스코드가 잘못되었다고 확신하지 않는 한 CGI 질문을 아파치 버그 데이터베이스에 올리면 절대로 안된다.

Comments

Notice:
This is not a Q&A section. Comments placed here should be pointed towards suggestions on improving the documentation or server, and may be removed by our moderators if they are either implemented or considered invalid/off-topic. Questions on how to manage the Apache HTTP Server should be directed at either our IRC channel, #httpd, on Libera.chat, or sent to our mailing lists.