jinyPHP 3권 - 12. CURL

**12. CURL**

====================

보통 웹페이지들은 브라우저를 통하여 접속합니다. 또한 브라우저는 전송받은 HTML을 분석하고 javascript를 실행한 결과를 화면에 그래픽 처리 결과로 보여 줍니다.

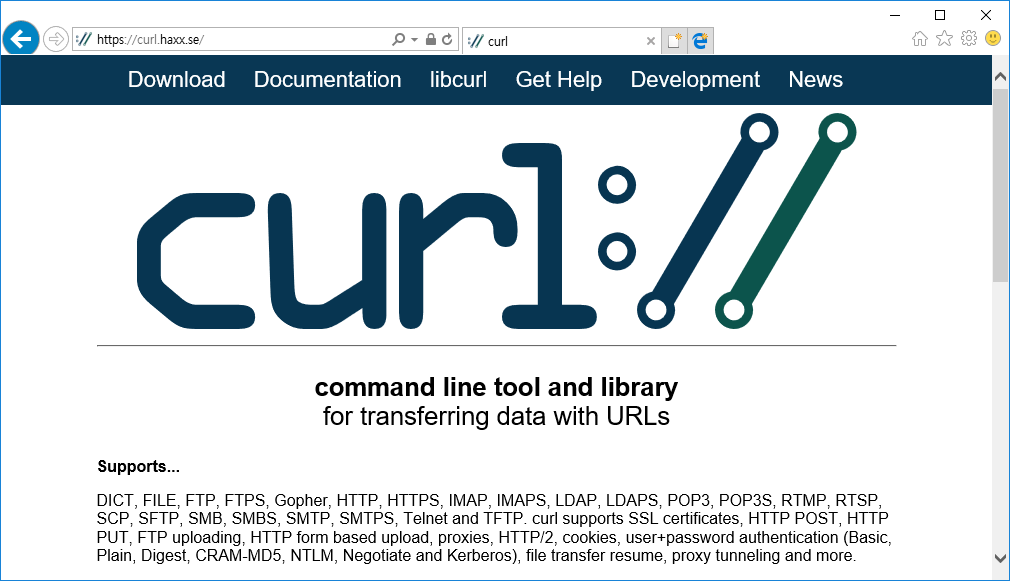
cURL은 Client URL의 약자 입니다. cURL은 command line tool로 콘솔창을 통하여 웹사이트에 접속하고 결과를 받아 올 수 있습니다. cURL은 다양한 프로토콜을 지원합니다. 그중 많이 사용하는 프로토콜로는 HTTP,FTP 등이 있습니다.

웹 URL을 접근할때 fopen() 함수등을 통하여 접근도 가능하나 보완적인 측면 때문에 접속을 제한합니다. 하지만 cURL은 PHP의 allow\_url\_fopen 옵션설정과 상관 없이 동작을 합니다. 또한 Proxy, Cookie, Header를 쉽게 세팅할 수 있습니다.

**12.1 설치**

====================

cURL 의 모듈, 라이브러리 및 설치파일들은 공식 사이트 https://curl.haxx.se/ 에서 다운로드 받을 수 있습니다.



**12.1.1 모듈 설정**

====================

cURL 확장모듈이 설정이 되어 있지 않는다면 php.ini 파일을 직접 수정을 해주어야 합니다.

확장 모듈의 설치 경로를 php가 설치된 디렉토리로 변경, 주석을 해제합니다.

**; On windows:**

**extension\_dir = "C:\php-7.1.4-Win32-VC14-x86\ext"**

확장모듈부분에서 cURL 모듈부분을 설정, 주석을 해제합니다.

**extension=php\_curl.dll**

php.ini를 변경후에 내장서버 또는 아파치 서버를 재시작 해주면 됩니다.

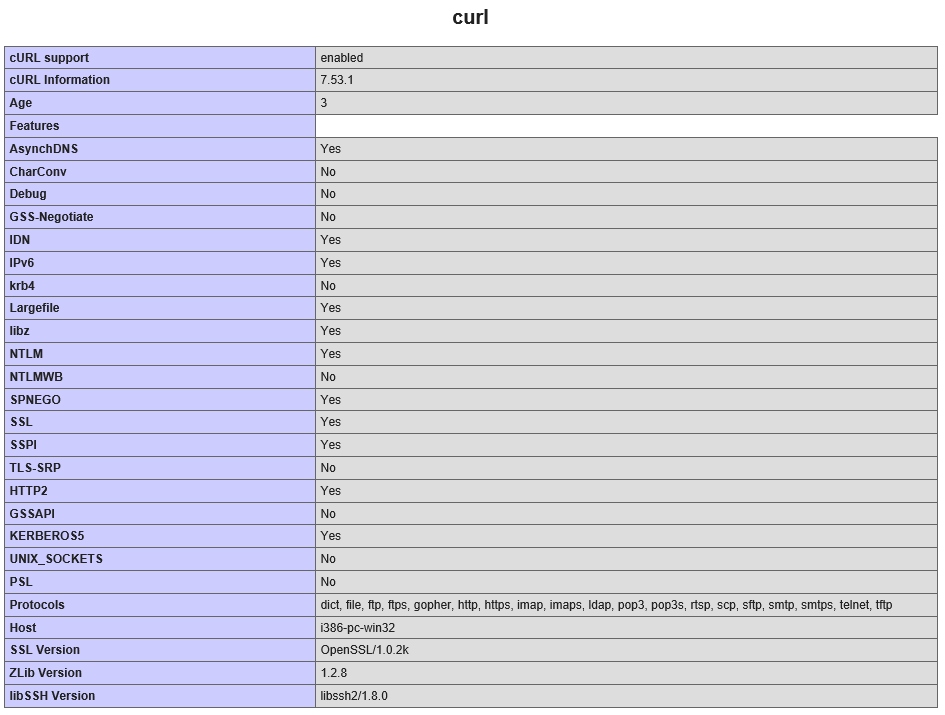
**12.1.2 모듈 확인**

====================

PHP는 코드 소스상에서 cURL 기능을 사용할 수 있는 내부함수들을 확장기능으로 제공합니다. 확장기능이 설치되어 있는 확인을 하는 방법은 콘솔상에서 명령어를 입력해 보면 쉽게 알 수 있습니다.

**#] php --re curl**

또는 phpinfo() 함수를 통해서 확인을 할 수 있습니다.

****

**12.1.3 모듈 체크**

====================

프로그램 실행시 cURL미설치로 발생될 수 있는 오류를 사전에 방지하기 위하여 소스상에서 확장모듈을 확인할 수 있습니다.

**예제소스) curl.php**

|  |
| --- |
| <?php  if(extension\_loaded("curl")){  echo "cURL extension is Loaded";  } else {  echo "cURL extension is not available";  }  ?> |

**화면출력)**

cURL extension is Loaded

**12.2 기본동작**

====================

cURL 확장모듈이 잘 설정이 되었다면 PHP curl 함수들을 사용할 수 있습니다.

**12.2.1 초기화**

====================

cURL을 사용하기 위해서는 먼저 cURL 세션의 초기화가 필요로 합니다. 세션초기화를 위해서는 curl\_init ()을 사용합니다.

**|내부함수|**

|  |
| --- |
| resource **curl\_init** ([ string $url = NULL ] ) |

$ch = curl\_init();

위와 같이 세션의 초기화 후에 curl\_setopt (), curl\_exec () 를 통하여 cURL 세션을 실행할 수 있습니다. 모든 cURL 세션 후에는 curl\_close ()를 통하여 세션을 종료합니다.

**예제) curl\_basic.php**

|  |
| --- |
| <?php  $ch = curl\_init("http://www.example.com/");  $fp = fopen("example.txt", "w");  curl\_setopt($ch, CURLOPT\_FILE, $fp);  curl\_setopt($ch, CURLOPT\_HEADER, 0);  curl\_exec($ch);  curl\_close($ch);  fclose($fp);  ?> |

위의 예제는 지정한 웹사이트 URL로 접속하여 HTML 페이지를 받아 지정한 파일로 저장을 하는 예제 입니다.

**12.2.2 전송옵션**

====================

**|내부함수|**

|  |
| --- |
| bool **curl\_setopt** ( resource $ch , int $option , mixed $value ) |

내부함수 curl\_setopt()는 cURL 전송을 위한 옵션을 설정 설정합니다. 옵션은 CURLOPT\_\*\*\* 형태의 상수값으로 되어 있습니다.

**|내부함수|**

|  |
| --- |
| bool **curl\_setopt\_array** ( resource $ch , array $options ) |

내부함수 curl\_setopt\_array()는 CURL 전송을 위한 옵션을 배열로 설정할 수 있습니다.

|  |
| --- |
| $options = array(  CURLOPT\_URL => 'http://www.example.com/',  CURLOPT\_HEADER => false  );  curl\_setopt\_array($ch, $options); |

curl\_setopt\_array() 함수를 사용시 앞전의 curl\_setopt() 함수를 통하여 여러번 호출을 하지 않고 옵션을 배열그룹으로 전달 할 수 있습니다.

**|내부함수|**

|  |
| --- |
| void **curl\_reset** ( resource $ch ) |

내부함수 curl\_reset()는 설정한 cURL 세션의 설정값을 리셋합니다.

|  |
| --- |
| $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL, 'http://aaa.com/');  // 옵션 설정을 모두 리셋  curl\_reset($ch); |

**12.2.3 전송 실행**

====================

설정한 cURL 세션값등을 실행하여 통신을 처리를 합니다. cURL 세션을 실행하기 위해서는 내부함수curl\_exec()를 사용합니다.

**|내부함수|**

|  |
| --- |
| mixed **curl\_exec** ( resource $ch ) |

실행 성공시 결과값을 반환합니다. 반환된 값을 받아 처리를 합니다.

|  |
| --- |
| $response = curl\_exec($ch); |

**12.2.4 세션 종료**

====================

cURL 생성과 실행이 완료가 된 경우에는 세션을 종료해 줍니다. 내부함수 curl\_close()는 cURL 세션을 종료합니다.

**|내부함수|**

|  |
| --- |
| void **curl\_close** ( resource $ch ) |

|  |
| --- |
| curl\_close($ch); |

**12.3 공유핸들**

====================

공유핸들은 다수의 cURL 핸들에게 동일한 설정값을 적용한 핸들을 이용하여 설정값을 지정할 수 있는 기능입니다.

**|내부함수|**

|  |
| --- |
| resource **curl\_share\_init** ( void ) |

내부함수 curl\_share\_init()는공유 cURL 세션핸들을 초기화 합니다.

**|내부함수|**

|  |
| --- |
| bool **curl\_share\_setopt** ( resource $sh , int $option , string $value ) |

내부함수 curl\_share\_setopt()는공유 cURL 핸들에 옵션값을 설정합니다.

**|내부함수|**

|  |
| --- |
| void **curl\_share\_close** ( resource $sh ) |

내부함수 curl\_share\_close()는공유 cURL 핸들을 종료합니다.

**예제) curl\_share.php**

|  |
| --- |
| <?php  // cURL 공유 핸드를 초기화 합니다.  // 쿠기값과 같은 공유 옵션을 설정합니다.  $sh = curl\_share\_init();  curl\_share\_setopt($sh, CURLSHOPT\_SHARE, CURL\_LOCK\_DATA\_COOKIE);  $ch1 = curl\_init("http://example.com/");  // 공유 핸들의 옵션을 통하여 ch1의 curl 핸들 옵션을 설정합니다.  curl\_setopt($ch1, CURLOPT\_SHARE, $sh);  curl\_exec($ch1);  $ch2 = curl\_init("http://php.net/");  // 공유 핸들의 옵션을 통하여 ch2의 curl 핸들 옵션을 설정합니다.  curl\_setopt($ch2, CURLOPT\_SHARE, $sh);  curl\_exec($ch2);  // 공유 핸들을 종료 합니다.  curl\_share\_close($sh);  curl\_close($ch1);  curl\_close($ch2);  ?> |

**12.4 멀티핸들**

====================

멀티 CURL 처리를 위한 함수들을 지원합니다. 멀티핸들은 한 개의 작업별로 작업하던 cURL 작업 대신에 여러 개를 작업을 할 수 있도록 설정하고 실행하는 방법입니다.

**|내부함수|**

|  |
| --- |
| resource **curl\_multi\_init** ( void ) |

내부함수 curl\_multi\_init()는 멀티 핸들을 초기화 합니다. 멀티핸들은 여러개의 cURL 핸들을 비동기 방식으로 처리 할 수 ​​있습니다.

**|내부함수|**

|  |
| --- |
| int **curl\_multi\_add\_handle** ( resource $mh , resource $ch ) |

내부함수 curl\_multi\_add\_handle()는 단일 cURL 핸들을 멀티 cURL 핸들에 추가합니다.

**|내부함수|**

|  |
| --- |
| int **curl\_multi\_exec** ( resource $mh , int &$still\_running ) |

내부함수 curl\_multi\_exec()는 멀티 cURL 핸들에 속해있는 하위 핸들을 실행합니다. 첫번째 인자는 멀티 핸들, 두번째 인자는 작업이 실행 도중인지 여부를 알려주는 플래그 참조값 입니다.

**|내부함수|**

|  |
| --- |
| int **curl\_multi\_select** ( resource $mh [, float $timeout = 1.0 ] ) |

내부함수 curl\_multi\_select()는 모든 curl\_multi 연결에 대한 작업을 대기합니다. curl\_multi 연결 중에서 한개의 활동이있을 때까지 차단합니다.

성공시, 디스크립터 세트에 포함 된 디스크립터의 수를 반환합니다. 활동이 없으면 0 을 반환하고 실패하면 -1을 반환합니다.

**|내부함수|**

|  |
| --- |
| int **curl\_multi\_remove\_handle** ( resource $mh , resource $ch ) |

내부함수 curl\_multi\_remove\_handle()는 멀티 핸들에서 단일 핸들을 제거합니다.

**|내부함수|**

|  |
| --- |
| void **curl\_multi\_close** ( resource $mh ) |

내부함수 curl\_multi\_close()는 멀티 cURL 핸들을 종료합니다.

**예제) curl\_multi.php**

|  |
| --- |
| <?php    // cURL 단일 핸들을 초기화 합니다.  $ch1 = curl\_init();  // 각각의 핸들에 옵션값을 설정합니다.  $url1 = "http://www.example.com/";  curl\_setopt($ch1, CURLOPT\_URL, $url1);  curl\_setopt($ch1, CURLOPT\_HEADER, 0);  // cURL 단일 핸들을 초기화 합니다.  $ch2 = curl\_init();  // 각각의 핸들에 옵션값을 설정합니다.  $url2 = "http://www.example.com/";  curl\_setopt($ch2, CURLOPT\_URL, $url2 );  curl\_setopt($ch2, CURLOPT\_HEADER, 0);  //멀티 핸들을 생성합니다.  $mh = curl\_multi\_init();  //멀티 핸들에 단일핸들 들을 추가합니다.  curl\_multi\_add\_handle($mh,$ch1);  curl\_multi\_add\_handle($mh,$ch2);  // 멀티 실행 여부 체크 플레그  $active = null;    do {  // 멀티 핸들을 실행합니다.  $mrc = curl\_multi\_exec($mh, $active);  } while ($mrc == CURLM\_CALL\_MULTI\_PERFORM);  while ($active && $mrc == CURLM\_OK) {    // curl\_multi 연결에 대한 작업을 대기합니다.  // 실패시 -1을 반환  if (curl\_multi\_select($multi) == -1) {  // 마이크로 초 지연 실행  usleep(1);  } else {  do {  // 멀티 핸들을 실행합니다.  $mrc = curl\_multi\_exec($mh, $active);  } while ($mrc == CURLM\_CALL\_MULTI\_PERFORM);    }    }  // 멀티 핸들에서 단일 핸들을 제거합니다.  curl\_multi\_remove\_handle($mh, $ch1);  curl\_multi\_remove\_handle($mh, $ch2);    // 멀티 cURL 핸들을 종료합니다.  curl\_multi\_close($mh);    ?> |

**|내부함수|**

|  |
| --- |
| string **curl\_multi\_getcontent** ( resource $ch ) |

내부함수 curl\_multi\_getcontent()는 CURLOPT\_RETURNTRANSFER가 설정된 경우 cURL 핸들의 내용을 반환합니다.

CURLOPT\_RETURNTRANSFER가 특정 핸들에 대해 설정된 옵션 인 경우에 cURL 핸들의 내용을 문자열 형식으로 반환합니다.

**예제) curl\_multi\_getcontent.php**

|  |
| --- |
| <?php  $ch = curl\_init('http://www.example.com/');  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);  // curl\_exec()로 실행 된 핸들에 curl\_multi\_getcontent()를 사용할 수 있습니다.  $result = curl\_exec($ch);  // curl\_multi\_getcontent()는 curl\_exec ()와 같은 결과값을 반환합니다.  $content = curl\_multi\_getcontent($ch);  var\_dump($result === $content);  echo $content;    curl\_close($ch);  ?> |

**|내부함수|**

|  |
| --- |
| bool **curl\_multi\_setopt** ( resource $mh , int $option , mixed $value ) |

내부함수 curl\_multi\_setopt()는 cURL 멀티 핸들에 대한 옵션을 설정합니다.

**|내부함수|**

|  |
| --- |
| array **curl\_multi\_info\_read** ( resource $mh [, int &$msgs\_in\_queue = NULL ] ) |

내부함수 curl\_multi\_info\_read()는 현재 전송의 정보를 읽어 옵니다.

각각의 전송에서 메시지 또는 정보가 있는 경우에는 멀티 핸들로 요청을 하여야 합니다. 메시지는 전송 오류 코드 또는 전송이 완료되었다는 정보 만 표시됩니다.

만일 이 함수를 반복적으로 호출할때는 매번 새로운 다른 결과값이 반환 될 수 있습니다. FALSE 상태로 더 이상 반환되는 값을 얻을 수 없을 때까지 반복 됩니다. msgs\_in\_queue가 가리키는 정수에는이 함수가 호출 된 이후 남은 메시지 수가 포함됩니다.

**예제) curl\_multi\_info\_read.php**

|  |
| --- |
| <?php  $urls = array(  "http://www.exsample.com/",  "http://www.php.net/"  );  $mh = curl\_multi\_init();  foreach ($urls as $i => $url) {  $conn[$i] = curl\_init($url);  curl\_setopt($conn[$i], CURLOPT\_RETURNTRANSFER, 1);  curl\_multi\_add\_handle($mh, $conn[$i]);  }  do {  $status = curl\_multi\_exec($mh, $active);  $info = curl\_multi\_info\_read($mh);  if (false !== $info) {  var\_dump($info);  echo "<br>";  }  } while ($status === CURLM\_CALL\_MULTI\_PERFORM || $active);  foreach ($urls as $i => $url) {  $res[$i] = curl\_multi\_getcontent($conn[$i]);  curl\_close($conn[$i]);  }  var\_dump(curl\_multi\_info\_read($mh));  ?> |

**화면출력)**

array(3) { ["msg"]=> int(1) ["result"]=> int(0) ["handle"]=> resource(4) of type (curl) }

array(3) { ["msg"]=> int(1) ["result"]=> int(52) ["handle"]=> resource(3) of type (curl) }

bool(false)

**|내부함수|**

|  |
| --- |
| string **curl\_multi\_strerror** ( int $errornum ) |

내부함수 curl\_multi\_strerror()는 오류 코드를 설명하는 텍스트 오류 메시지를 리턴합니다.

**예제) curl\_multi\_strerror.php**

|  |
| --- |
| <?php  $ch1 = curl\_init("http://example.com");  $ch2 = curl\_init("http://www.php.net/");  $mh = curl\_multi\_init();  curl\_multi\_add\_handle($mh, $ch1);  curl\_multi\_add\_handle($mh, $ch2);  // 멀티 실행 여부 체크 플레그  $active = null;  do {  $status = curl\_multi\_exec($mh, $active);  // Check for errors  if($status > 0) {  // Display error message  echo "ERROR!\n " . curl\_multi\_strerror($status);  }  } while ($status === CURLM\_CALL\_MULTI\_PERFORM || $active);  ?> |

**12.5 cURL 오류처리**

====================

통신을 처리할때는 예외적인 상황들이 자주 발생을 합니다. cURL을 처리할 때 발생될 수 있는 오류들을 처리할 수 있는 몇가지 함수들을 지원합니다.

**|내부함수|**

|  |
| --- |
| string **curl\_error** ( resource $ch ) |

내부함수 curl\_error()는 cURL 오류가 발생된 경우 에러 문자열을 읽어 옵니다.

**|내부함수|**

|  |
| --- |
| int **curl\_errno** ( resource $ch ) |

내부함수 curl\_errno()는 오류 코드를 반환합니다.

**|내부함수|**

|  |
| --- |
| string **curl\_strerror** ( int $errornum ) |

내부함수 curl\_strerror()는 오류 코드의 문자열을 반환합니다.

**예제) curl\_error.php**

|  |
| --- |
| <?php  // 존재하지 않은 url의 세션을 초기화 합니다.  $ch = curl\_init('http://404.php.net/');  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);  if (curl\_exec($ch) === false)  {  // 오류 발생시 : 오류 메시지를 읽어옵니다.  echo "Curl error Message : ". curl\_error($ch) ."<br>";    // 오류 넘버를 출력합니다.  $errNo = curl\_errno ($ch);  echo "cURL Error No : ". $errNo;  echo " == ". curl\_strerror($errNo);  } else {  echo 'success';  }  // 종료  curl\_close($ch);  ?> |

**화면출력)**

Curl error Message : Could not resolve host: 404.php.net

cURL Error No : 6 == Couldn't resolve host name

**12.6 cURL 그외함수**

====================

그외 CURL을 처리하는 도움이 되는 몇 개의 관련 함수를 지원합니다.

**|내부함수|**

|  |
| --- |
| array **curl\_version** ([ int $age = CURLVERSION\_NOW ] ) |

내부함수 curl\_version()는 cURL 버전 정보를 가지고 옵니다. 버전 정보를 배열값으로 반환합니다.

**예제) curl\_version.php**

|  |
| --- |
| <?php  $version = curl\_version();  print\_r($version);  ?> |

**화면출력)**

Array ( [version\_number] => 472321 [age] => 3 [features] => 2428829 [ssl\_version\_number] => 0 [version] => 7.53.1 [host] => i386-pc-win32 [ssl\_version] => OpenSSL/1.0.2k [libz\_version] => 1.2.8 [protocols] => Array ( [0] => dict [1] => file [2] => ftp [3] => ftps [4] => gopher [5] => http [6] => https [7] => imap [8] => imaps [9] => ldap [10] => pop3 [11] => pop3s [12] => rtsp [13] => scp [14] => sftp [15] => smtp [16] => smtps [17] => telnet [18] => tftp ) )

**|내부함수|**

|  |
| --- |
| string **curl\_escape** ( resource $ch , string $str ) |

내부함수 curl\_escape()는 주어진 문자열을 URL을 RFC 3986에 따라 인코딩합니다.

**예제) curl\_escape.php**

|  |
| --- |
| <?php  $ch = curl\_init();  // GET 파라미터로 사용되는 문자열 값을 RFC3986 인코딩 합니다.  $location = curl\_escape($ch, 'aaa / bbb');  $url = "http://example.com/test.php?location={$location}";  echo $url;    ?> |

**화면출력)**

http://example.com/test.php?location=aaa%20%2F%20bbb

**|내부함수|**

|  |
| --- |
| string **curl\_unescape** ( resource $ch , string $str ) |

내부함수 curl\_unescape()는 URL을 RFC 3986에 따라 디코딩합니다.

**예제) curl\_unescape.php**

|  |
| --- |
| <?php  $ch = curl\_init();  // GET 파라미터로 사용되는 문자열 값을 RFC3986 인코딩 합니다.  $location = curl\_escape($ch, 'aaa / bbb');  $url = "http://example.com/test.php?location={$location}";  echo $url . "<br>";  echo curl\_unescape($ch , $location);  ?> |

**화면출력)**

http://example.com/test.php?location=aaa%20%2F%20bbb

aaa / bbb

**|내부함수|**

|  |
| --- |
| mixed **curl\_getinfo** ( resource $ch [, int $opt ] ) |

내부함수 curl\_getinfo()는 cURL 세션실행 성공시 마지막 전송에 관련된 정보를 얻습니다.

* **CURLINFO\_EFFECTIVE\_URL** - 마지막 유효 URL
* CURLINFO\_HTTP\_CODE - 마지막으로 수신 한 HTTP 코드
* CURLINFO\_FILETIME - 검색된 문서의 원격 시간 (CURLOPT\_FILETIME이 활성화 됨). -1이 반환되면 문서의 시간을 알 수 없습니다.
* CURLINFO\_TOTAL\_TIME - 마지막 전송에 대한 총 트랜잭션 시간 (초)
* CURLINFO\_NAMELOOKUP\_TIME - 이름 확인이 완료 될 때까지의 시간 (초)
* CURLINFO\_CONNECT\_TIME - 연결을 설정하는 데 걸리는 시간 (초)
* CURLINFO\_PRETRANSFER\_TIME - 시작부터 파일 전송이 시작되기 직전까지의 시간 (초)
* CURLINFO\_STARTTRANSFER\_TIME - 첫 번째 바이트가 전송 될 때까지의 초 단위 시간
* CURLINFO\_REDIRECT\_COUNT - CURLOPT\_FOLLOWLOCATION 옵션을 사용하는 리디렉션의 수
* CURLINFO\_REDIRECT\_TIME - CURLOPT\_FOLLOWLOCATION 옵션을 사용하여 최종 트랜잭션이 시작되기 전의 모든 리디렉션 단계의 시간 (초)
* CURLINFO\_REDIRECT\_URL - CURLOPT\_FOLLOWLOCATION 옵션을 사용 중지 한 경우 : 마지막 트랜잭션에서 찾은 리디렉션 URL을 수동으로 요청해야합니다. CURLOPT\_FOLLOWLOCATION 옵션을 사용하면 비어 있습니다. 이 경우 리디렉션 URL은 CURLINFO\_EFFECTIVE\_URL에서 사용할 수 있습니다.
* CURLINFO\_PRIMARY\_IP - 가장 최근 연결의 IP 주소
* CURLINFO\_PRIMARY\_PORT - 가장 최근 연결의 대상 포트
* CURLINFO\_LOCAL\_IP - 가장 최근 연결의 로컬 (소스) IP 주소
* CURLINFO\_LOCAL\_PORT - 가장 최근 연결의 로컬 (소스) 포트
* CURLINFO\_SIZE\_UPLOAD - 업로드 된 총 바이트 수
* CURLINFO\_SIZE\_DOWNLOAD - 다운로드 된 총 바이트 수
* CURLINFO\_SPEED\_DOWNLOAD - 평균 다운로드 속도
* CURLINFO\_SPEED\_UPLOAD - 평균 업로드 속도
* CURLINFO\_HEADER\_SIZE - 수신 된 모든 헤더의 총 크기
* CURLINFO\_HEADER\_OUT - 요청 문자열이 전송되었습니다. 이 작업을 수행하려면 curl\_setopt ()를 호출하여 핸들에 CURLINFO\_HEADER\_OUT 옵션을 추가하십시오.
* CURLINFO\_REQUEST\_SIZE - 현재 HTTP 요청에 대해서만 발행 된 요청의 총 크기
* CURLINFO\_SSL\_VERIFYRESULT - CURLOPT\_SSL\_VERIFYPEER를 설정하여 요청한 SSL 인증 확인 결과
* CURLINFO\_CONTENT\_LENGTH\_DOWNLOAD - 다운로드 길이, Content-Length : 필드에서 읽음
* CURLINFO\_CONTENT\_LENGTH\_UPLOAD - 지정된 업로드 크기
* CURLINFO\_CONTENT\_TYPE - Content-Type : 요청한 문서의 이름입니다. NULL은 서버가 유효한 Content-Type : 헤더를 보내지 않았 음을 나타냅니다.
* CURLINFO\_PRIVATE - 이전에 curl\_setopt ()의 CURLOPT\_PRIVATE 옵션으로 설정 한이 cURL 핸들과 연관된 개인 데이터.
* CURLINFO\_RESPONSE\_CODE - 마지막 응답 코드
* CURLINFO\_HTTP\_CONNECTCODE - CONNECT 응답 코드
* CURLINFO\_HTTPAUTH\_AVAIL - 이전 응답에 따라 사용 가능한 인증 방법을 나타내는 비트 마스크
* CURLINFO\_PROXYAUTH\_AVAIL - 이전 응답에 따라 사용 가능한 프록시 인증 방법을 나타내는 비트 마스크
* CURLINFO\_OS\_ERRNO - 연결 실패로 인한 Errno입니다. 번호는 OS 및 시스템에 따라 다릅니다.
* CURLINFO\_NUM\_CONNECTS - 이전 전송을 위해 컬이 작성해야하는 연결 수
* CURLINFO\_SSL\_ENGINES - OpenSSL 암호화 엔진 지원
* CURLINFO\_COOKIELIST - 모든 알려진 쿠키
* CURLINFO\_FTP\_ENTRY\_PATH - FTP 서버의 입력 경로
* CURLINFO\_APPCONNECT\_TIME - 시작부터 SSL / SSH 연결 / 핸드 셰이크가 원격 호스트에 완료 될 때까지 걸린 시간 (초)
* CURLINFO\_CERTINFO - TLS 인증서 체인
* CURLINFO\_CONDITION\_UNMET - 충족되지 않은 시간 조건에 대한 정보
* CURLINFO\_RTSP\_CLIENT\_CSEQ - 다음 RTSP 클라이언트 CSeq
* CURLINFO\_RTSP\_CSEQ\_RECV - 최근받은 CSeq
* CURLINFO\_RTSP\_SERVER\_CSEQ - 다음 RTSP 서버 CSeq
* CURLINFO\_RTSP\_SESSION\_ID - RTSP 세션 ID

**예제) curl\_getinfo.php**

|  |
| --- |
| <?php  $ch = curl\_init('http://www.example.com/');  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);  if ($response = curl\_exec($ch)) {    $info = curl\_getinfo($ch);  print\_r($info);  echo "<br>";  switch ($http\_code = curl\_getinfo($ch, CURLINFO\_HTTP\_CODE)) {  case 200:  echo 'success <br>';  break;  default:  echo 'Unexpected HTTP code: ', $http\_code, "\n";  }  } else {  // 오류 발생시 : 오류 메시지를 읽어옵니다.  echo "Curl error Message : ". curl\_error($ch) ."<br>";    // 오류 넘버를 출력합니다.  echo "cURL Error No : ". curl\_errno ($ch);  }  // 종료  curl\_close($ch);  ?> |

**화면출력)**

Array ( [url] => http://www.example.com/ [content\_type] => text/html [http\_code] => 200 [header\_size] => 311 [request\_size] => 54 [filetime] => -1 [ssl\_verify\_result] => 0 [redirect\_count] => 0 [total\_time] => 0.312 [namelookup\_time] => 0 [connect\_time] => 0.156 [pretransfer\_time] => 0.156 [size\_upload] => 0 [size\_download] => 1270 [speed\_download] => 4070 [speed\_upload] => 0 [download\_content\_length] => 1270 [upload\_content\_length] => -1 [starttransfer\_time] => 0.312 [redirect\_time] => 0 [redirect\_url] => [primary\_ip] => 93.184.216.34 [certinfo] => Array ( ) [primary\_port] => 80 [local\_ip] => 172.30.1.3 [local\_port] => 57793 )

success

**|내부함수|**

|  |
| --- |
| resource **curl\_copy\_handle** ( resource $ch ) |

내부함수 curl\_copy\_handle()는 cURL 핸들을 환경 설정과 함께 복사합니다.

**예제) curl\_copy\_handle.php**

|  |
| --- |
| <?php  $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL, 'http://www.example.com/');  curl\_setopt($ch, CURLOPT\_HEADER, 0);  // cURL handle을 복사합니다.  $ch2 = curl\_copy\_handle($ch);  curl\_exec($ch2);  curl\_close($ch2);  curl\_close($ch);  ?> |

**12.7 POST 접속응용**

====================

cURL의 기본 기능들을 이용하여 해당 사이트에 POST 형태로 접속할 수 있습니다. 다음은 POST 방식으로 웹 URL을 읽어오는 메소드 함수의 예제입니다.

|  |
| --- |
| public function \_curl\_post($url, $postfild){  $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL,$url);  curl\_setopt($ch, CURLOPT\_POST, 1);  curl\_setopt($ch, CURLOPT\_POSTFIELDS,$postfild);  // receive server response ...  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);  $server\_output = curl\_exec($ch);  curl\_close($ch);  return $server\_output;  } |

**12.8 파일 업로드**

====================

cURL을 응용하여 POST 파일 업로드도 가능합니다. 다음은 curl을 이용하여 파일을 업로드 하는 예제 함수 입니다.

|  |
| --- |
| function curl\_upload($mode, $target\_url, $path, $file){  $file\_name\_with\_full\_path = realpath($file);  $post = array('mode'=>$mode, 'path'=>$path, 'file\_contents'=>'@'.$file\_name\_with\_full\_path);    $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL,$target\_url);  curl\_setopt($ch, CURLOPT\_POST,1);  curl\_setopt($ch, CURLOPT\_POSTFIELDS, $post);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER,1);  $result=curl\_exec($ch);  curl\_close($ch);    return $result;  } |