Chapter

12

cURL

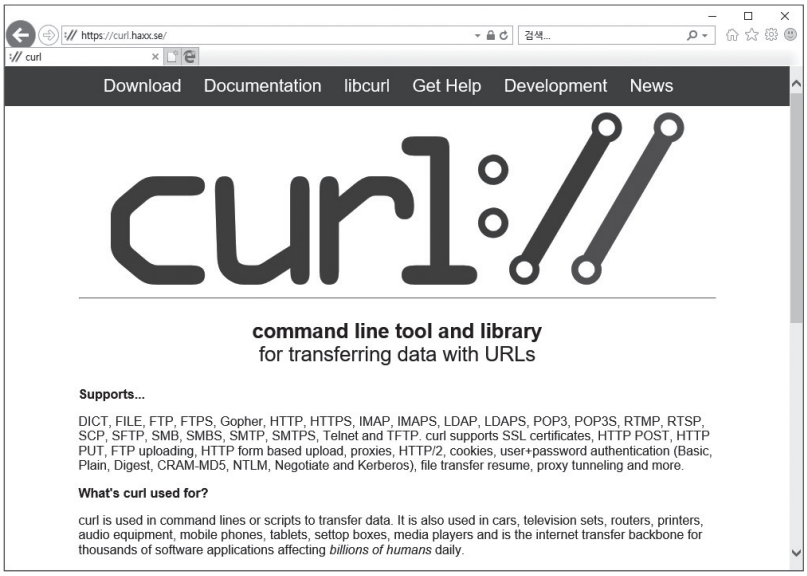
보통 웹 페이지들은 브라우저를 통하여 접속합니다. 또한 브라우저는 전송받은 HTML을 분석하고 자바스크립트를 실행한 결과를 화면에 그래픽 처리 결과로 보여줍니다.

cURL은 client URL의 약자입니다. cURL은 command line tool로 콘솔 창을 통하여 웹 사이트에 접속하고 결과를 받아올 수 있습니다. cURL은 다양한 프로토콜을 지원합니다. 그중 많이 사용하는 프로토콜로는 HTTP, FTP 등이 있습니다.

웹 URL을 접근할 때 fopen() 함수 등을 통하여 접근도 가능하나, 보안적인 측면 때문에 접속을 제한합니다. 하지만 cURL은 PHP의 allow_url_fopen 옵션 설정과 상관없이 동작합니다. 또한 Proxy, Cookie, Header를 쉽게 설정할 수 있습니다.

12.1 설치

cURL의 모듈, 라이브러리 및 설치 파일들은 공식 사이트 https://curl.haxx.se/에서 다운로드할 수 있습니다.



12.1.1 모듈 설정

cURL 확장 모듈이 설정되어 있지 않는다면 php.ini 파일을 직접 수정해야 합니다.

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

```
; On windows:
extension_dir = "C:₩php-7.1.4-Win32-VC14-x86₩ext"
```

확장 모듈 부분에서 cURL 모듈 부분을 설정, 주석을 해제합니다.

```
extension=php_curl.dll
```

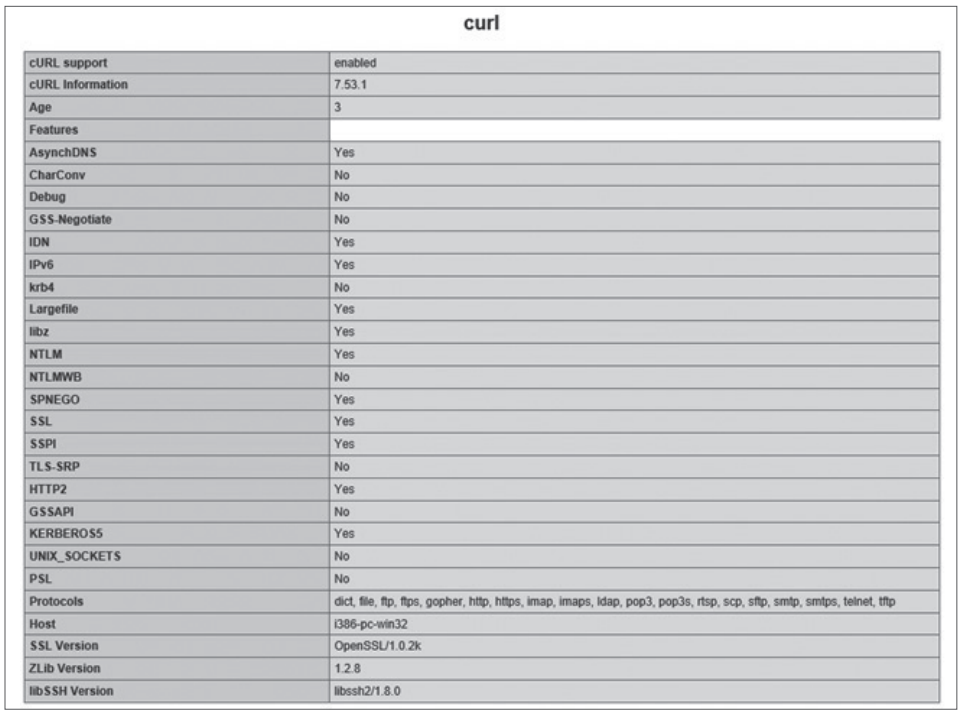
php.ini를 변경 후에 내장 서버 또는 아파치 서버를 재시작하면 됩니다.

12.1.2 모듈 확인

PHP는 코드 소스상에서 cURL 기능을 사용할 수 있는 내장 함수들을 확장 기능으로 제공합니다. 확장 기능이 설치되어 있는지 확인하는 방법은 콘솔상에서 명령어를 입력해 보면 쉽게 알 수 있습니다.

#] php --re curl

또는 phpinfo() 함수를 통해서 확인할 수 있습니다.



curl

cURL support	enabled
cURL Information	7.53.1
Age	3
Features	
AsynchDNS	Yes
CharConv	No
Debug	No
GSS-Negotiate	No
IDN	Yes
IPv6	Yes
krb4	No
Largefile	Yes
libz	Yes
NTLM	Yes
NTLMWB	No
SPNEGO	Yes
SSL	Yes
SSPI	Yes
TLS-SRP	No
HTTP2	Yes
GSSAPI	No
KERBEROS5	Yes
UNIX_SOCKETS	No
PSL	No
Protocols	dict, file, ftp, ftps, gopher, http, https, imap, imaps, ldap, pop3, pop3s, rtsp, scp, sftp, smtp, smtps, telnet, tftp
Host	i386-pc-win32
SSL Version	OpenSSL/1.0.2k
ZLib Version	1.2.8
libSSH Version	libssh2/1.8.0

12.1.3 모듈 체크

프로그램 실행 시 cURL 미설치로 발생할 수 있는 오류를 사전에 방지하기 위하여 소스상에서 확장 모듈을 확인할 수 있습니다.

예제 파일 | curl.php

```
1  <?php
2     if (extension_loaded("curl")) {
3        echo "cURL extension is Loaded";
4     } else {
5        echo "cURL extension is not available";
6     }
7  ?>
```

화면 출력

```
cURL extension is Loaded
```

12.2 기본 동작

cURL 확장 모듈이 잘 설정되었다면 PHP cURL 함수들을 사용할 수 있습니다.

12.2.1 초기화

cURL을 사용하기 위해서는 먼저 cURL 세션의 초기화가 필요합니다. 세션 초기화를 위해서는 curl_init()를 사용합니다.

| 내장 함수 |

```
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 파일 | curl_basic.php

```
1   <?php
2
3       $ch = curl_init("http://www.example.com/");
4       $fp = fopen("example.txt", "w");
5
6       curl_setopt($ch, CURLOPT_FILE, $fp);
7       curl_setopt($ch, CURLOPT_HEADER, 0);
8
9       curl_exec($ch);
10      curl_close($ch);
11      fclose($fp);
```

```
12
13  ?>
```

위의 예제는 지정한 웹 사이트 URL로 접속하여 HTML 페이지를 받아 지정한 파일로 저장하는 예제입니다.

12.2.2 전송 옵션

| 내장 함수 |

```
bool curl_setopt ( resource $ch , int $option , mixed $value )
```

내장 함수 curl_setopt()는 cURL 전송을 위한 옵션을 설정합니다. 옵션은 CURLOPT_*** 형태의 상수값으로 되어 있습니다.

| 내장 함수 |

```
bool curl_setopt_array ( resource $ch , array $options )
```

내장 함수 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 사용 시 앞의 curl_setopt() 함수를 통하여 여러 번 호출하지 않고 옵션을 배열 그룹으로 전달할 수 있습니다.

| 내장 함수 |

```
void curl_reset ( resource $ch )
```

내장 함수 curl_reset()은 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 세션 값 등을 실행하여 통신을 처리합니다. cURL 세션을 실행하기 위해서는 내장 함수 curl_exec()를 사용합니다.

| 내장 함수 |

```
mixed curl_exec ( resource $ch )
```

실행 성공 시 결과값을 반환합니다. 반환된 값을 받아 처리합니다.

```
$response = curl_exec($ch);
```

12.2.4 세션 종료

cURL 생성과 실행이 완료된 경우에는 세션을 종료합니다. 내장 함수 curl_close()는 cURL 세션을 종료합니다.

| 내장 함수 |

```
void curl_close ( resource $ch )
```

```
curl_close($ch);
```

12.3 공유 핸들

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

| 내장 함수 |

```
resource curl_share_init ( void )
```

내장 함수 curl_share_init()는 공유 cURL 세션 핸들을 초기화합니다.

| 내장 함수 |

```
bool curl_share_setopt ( resource $sh , int $option , string $value )
```

내장 함수 curl_share_setopt()는 공유 cURL 핸들에 옵션값을 설정합니다.

| 내장 함수 |

```
void curl_share_close ( resource $sh )
```

내장 함수 curl_share_close()는 공유 cURL 핸들을 종료합니다.

예제 파일 | curl_share.php

```php
<?php
    // cURL 공유 핸들을 초기화합니다.
    // 쿠키 값과 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 종료합니다.
    curl_share_close($sh);

    curl_close($ch1);
    curl_close($ch2);

?>
```

12.4 멀티 핸들

멀티 CURL 처리를 위한 함수들을 지원합니다. 멀티 핸들은 1개의 작업별로 작업하던 cURL 작업 대신에 여러 개를 작업할 수 있도록 설정하고 실행하는 방법입니다.

| 내장 함수 |

```
resource curl_multi_init ( void )
```

내장 함수 curl_multi_init()는 멀티 핸들을 초기화합니다. 멀티 핸들은 여러 개의 cURL

핸들을 비동기 방식으로 처리할 수 있습니다.

| 내장 함수 |

```
int curl_multi_add_handle ( resource $mh , resource $ch )
```

내장 함수 curl_multi_add_handle()은 단일 cURL 핸들을 멀티 cURL 핸들에 추가합니다.

| 내장 함수 |

```
int curl_multi_exec ( resource $mh , int &$still_running )
```

내장 함수 curl_multi_exec()는 멀티 cURL 핸들에 속해 있는 하위 핸들을 실행합니다. 첫 번째 인자는 멀티 핸들, 두 번째 인자는 작업이 실행 도중인지 여부를 알려주는 플래그 참조 값입니다.

| 내장 함수 |

```
int curl_multi_select ( resource $mh [, float $timeout = 1.0 ] )
```

내장 함수 curl_multi_remove_handle()은 멀티 핸들에서 단일 핸들을 제거합니다.

| 내장 함수 |

```
void curl_multi_close ( resource $mh )
```

내장 함수 curl_multi_close()는 멀티 cURL 핸들을 종료합니다.

예제 파일 | **curl_multi.php**

```
1   <?php
```

```
2
3       // cURL 단일 핸들을 초기화합니다.
4       $ch1 = curl_init();
5       // 각각의 핸들에 옵션값을 설정합니다.
6       $url1 = "http://www.example.com/";
7       curl_setopt($ch1, CURLOPT_URL, $url1);
8       curl_setopt($ch1, CURLOPT_HEADER, 0);
9
10      // cURL 단일 핸들을 초기화합니다.
11      $ch2 = curl_init();
12      // 각각의 핸들에 옵션값을 설정합니다.
13      $url2 = "http://www.example.com/";
14      curl_setopt($ch2, CURLOPT_URL, $url2 );
15      curl_setopt($ch2, CURLOPT_HEADER, 0);
16
17      //멀티 핸들을 생성합니다.
18      $mh = curl_multi_init();
19
20      //멀티 핸들에 단일 핸들을 추가합니다.
21      curl_multi_add_handle($mh,$ch1);
22      curl_multi_add_handle($mh,$ch2);
23
24      // 멀티 실행 여부 체크 플래그
25      $active = null;
26
27      do {
28         // 멀티 핸들을 실행합니다.
29         $mrc = curl_multi_exec($mh, $active);
30      } while ($mrc == CURLM_CALL_MULTI_PERFORM);
31
32      while ($active && $mrc == CURLM_OK) {
33
34         // curl_multi 연결에 대한 작업을 대기합니다.
35         // 실패 시 -1을 반환
36         if (curl_multi_select($multi) == -1) {
37            // 마이크로 초 지연 실행
38            usleep(1);
39
40         } else {
41            do {
42               // 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);

?>
```

| 내장 함수 |

string **curl_multi_getcontent** (resource $ch)

내장 함수 curl_multi_getcontent()는 CURLOPT_RETURNTRANSFER가 설정된 경우 cURL 핸들의 내용을 반환합니다.

CURLOPT_RETURNTRANSFER가 특정 핸들에 대해 설정된 옵션 경우에 cURL 핸들의 내용을 문자열 형식으로 반환합니다.

예제 파일 | **curl_multi_getcontent.php**

```
<?php

    $ch = curl_init('http://www.example.com/');
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);

    // curl_exec()로 실행된 핸들에 curl_multi_getcontent()를 사용할 수 있습니다.
    $result = curl_exec($ch);

    // curl_multi_getcontent()는 curl_exec()와 같은 결과값을 반환합니다.
```

```
10      $content = curl_multi_getcontent($ch);
11
12      var_dump($result === $content);
13      echo $content;
14
15      curl_close($ch);
16
17  ?>
```

| 내장 함수 |

bool **curl_multi_setopt** (resource $mh , int $option , mixed $value)

내장 함수 curl_multi_setopt()는 cURL 멀티 핸들에 대한 옵션을 설정합니다.

| 내장 함수 |

array **curl_multi_info_read** (resource $mh [, int &$msgs_in_queue = NULL])

내장 함수 curl_multi_info_read()는 현재 전송의 정보를 읽어옵니다.

각각의 전송에서 메시지 또는 정보가 있는 경우에는 멀티 핸들로 요청해야 합니다. 메시지는 전송 오류 코드 또는 전송이 완료되었다는 정보만 표시됩니다.

만일 이 함수를 반복적으로 호출할 때는 매번 새로운 다른 결과값이 반환될 수 있습니다. FALSE 상태로 더 이상 반환되는 값을 얻을 수 없을 때까지 반복됩니다. msgs_in_queue가 가리키는 정수에는 이 함수가 호출 된 이후 남은 메시지 수가 포함됩니다.

예제 파일 | curl_multi_info_read.php

```
1   <?php
2
3       $urls = array(
4           "http://www.exsample.com/",
5            "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));

?>
```

화면 출력

```
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)
```

| 내장 함수 |

string **curl_multi_strerror** (int $errornum)

내장 함수 curl_multi_strerror()는 오류 코드를 설명하는 텍스트 오류 메시지를 반환합니다.

예제 파일 | curl_multi_strerror.php

```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 플래그
    $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 오류 처리

통신을 처리할 때는 예외적인 상황들이 자주 발생합니다. cURL을 처리할 때 발생될 수 있는 오류들을 처리할 수 있는 몇 가지 함수들을 지원합니다.

| 내장 함수 |

string **curl_error** (resource $ch)

내장 함수 curl_error()는 cURL 오류가 발생된 경우 에러 문자열을 읽어옵니다.

| 내장 함수 |

int **curl_errno** (resource $ch)

내장 함수 curl_errno()는 오류 코드를 반환합니다.

| 내장 함수 |

string **curl_strerror** (int $errornum)

내장 함수 curl_strerror()는 오류 코드의 문자열을 반환합니다.

예제 파일 | **curl_error.php**

```
1  <?php
2     // 존재하지 않은 url의 세션을 초기화합니다.
3     $ch = curl_init('http://404.php.net/');
4     curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
5
6     if (curl_exec($ch) === false)
7     {
8        // 오류 발생 시 : 오류 메시지를 읽어옵니다.
9        echo "Curl error Message : ". curl_error($ch) ."<br>";
10
11       // 오류 넘버를 출력합니다.
12       $errNo = curl_errno ($ch);
13       echo "cURL Error No : ". $errNo;
14
15       echo " == ". curl_strerror($errNo);
16
```

```
    } else {
        echo 'success';
    }

    // 종료
    curl_close($ch);

?>
```

화면 출력

```
Curl error Message : Could not resolve host: 404.php.net
cURL Error No : 6 == Couldn't resolve host name
```

12.6 cURL 그 외 함수

그 외 CURL을 처리하는 도움이 되는 몇 가지 관련 함수를 지원합니다.

| 내장 함수 |

array **curl_version** ([int $age = CURLVERSION_NOW])

내장 함수 curl_version()은 cURL 버전 정보를 가지고 옵니다. 버전 정보를 배열값으로 반환합니다.

예제 파일 | curl_version.php

```
<?php
    $version = curl_version();
    print_r($version);

?>
```

화면 출력

```
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 ) )
```

| 내장 함수 |

string **curl_escape** (resource $ch , string $str)

내장 함수 curl_escape()는 주어진 문자열을 URL을 RFC 3986에 따라 인코딩합니다.

예제 파일 | **curl_escape.php**

```php
<?php

    $ch = curl_init();

    // GET 파라미터로 사용되는 문자열 값을 RFC3986 인코딩합니다.
    $location = curl_escape($ch, 'aaa / bbb');

    $url = "http://example.com/test.php?location={$location}";
    echo $url;

?>
```

화면 출력

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

| 내장 함수 |

string **curl_unescape** (resource $ch , string $str)

내장 함수 curl_unescape()는 URL을 RFC 3986에 따라 디코딩합니다.

예제 파일 | curl_unescape.php

```php
<?php

    $ch = curl_init();

    // GET 파라미터로 사용되는 문자열 값을 RFC3986 인코딩합니다.
    $location = curl_escape($ch, 'aaa / bbb');

    $url = "http://example.com/test.php?location={$location}";
    echo $url . "<br>";

    echo curl_unescape($ch , $location);

?>
```

화면 출력

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

| 내장 함수 |

```
mixed curl_getinfo ( resource $ch [, int $opt ] )
```

내장 함수 curl_getinfo()는 cURL 세션 실행 성공 시 마지막 전송에 관련된 정보를 얻습니다.

- CURLINFO_EFFECTIVE_URL – 마지막 유효 URL
- CURLINFO_HTTP_CODE – 마지막으로 수신한 HTTP 코드
- CURLINFO_FILETIME – 검색된 문서의 원격 시간(CURLOPT_FILETIME이 활성화됨). –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 직전까지

의 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 중지한 경우: 마지막 트랜잭션에서 찾은 리디렉션 URL을 수동으로 요청해야 합니다. 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 – 업로드된 총 바이트 수
- CURLINFO_SIZE_DOWNLOAD – 다운로드된 총 바이트 수
- CURLINFO_SPEED_DOWNLOAD – 평균 다운로드 속도
- CURLINFO_SPEED_UPLOAD – 평균 업로드 속도
- CURLINFO_HEADER_SIZE – 수신된 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 발행된 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 않았음을 나타냅니다.
- CURLINFO_PRIVATE – 이전에 curl_setopt()의 CURLOPT_PRIVATE 옵션으로 설정한 이 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 프록 시 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 작성해야 하는 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 완료될 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 – 최근 받은 CSeq
- CURLINFO_RTSP_SERVER_CSEQ – 다음 RTSP 서버 CSeq
- CURLINFO_RTSP_SESSION_ID – RTSP 세션 ID

예제 파일 | **curl_getinfo.php**

```
1  <?php
2
3     $ch = curl_init('http://www.example.com/');
```

```
4       curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
5
6       if ($response = curl_exec($ch))   {
7
8          $info = curl_getinfo($ch);
9          print_r($info);
10         echo "<br>";
11
12         switch ($http_code = curl_getinfo($ch, CURLINFO_HTTP_CODE)) {
13            case 200:
14               echo 'success <br>';
15               break;
16            default:
17               echo 'Unexpected HTTP code: ', $http_code, "\n";
18         }
19
20      } else {
21         // 오류 발생 시 : 오류 메시지를 읽어옵니다.
22         echo "Curl error Message : ". curl_error($ch) ."<br>";
23
24         // 오류 넘버를 출력합니다.
25         echo "cURL Error No : ". curl_errno ($ch);
26
27      }
28
29      // 종료
30      curl_close($ch);
31  ?>
```

화면 출력

```
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
```

resource **curl_copy_handle** (resource $ch)

내장 함수 curl_copy_handle()은 cURL 핸들을 환경 설정과 함께 복사합니다.

예제 파일 | curl_copy_handle.php

```
1  <?php
2
3      $ch = curl_init();
4
5      curl_setopt($ch, CURLOPT_URL, 'http://www.example.com/');
6      curl_setopt($ch, CURLOPT_HEADER, 0);
7
8      // cURL handle을 복사합니다.
9      $ch2 = curl_copy_handle($ch);
10     curl_exec($ch2);
11
12     curl_close($ch2);
13     curl_close($ch);
14
15 ?>
```

12.7 POST 접속 응용

cURL의 기본 기능들을 이용하여 해당 사이트에 POST 형태로 접속할 수 있습니다. 다음은 POST 방식으로 웹 URL을 읽어오는 메서드 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 업로드하는 예제 함수입니다.

```
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;
}
```