Chapter

09

URL

09.1 네트워크 연결

09.1.1 IP 주소

IP 주소는 Internet Protocol의 약자입니다. 인터넷과 연결된 컴퓨터, 서버, 모바일 와이파이 등의 기기들은 상호 데이터를 전송하고 기기를 구분하기 위한 IP 주소를 할당합니다.

IP 주소는 컴퓨터, 모바일 기기들이 네트워크에 접속 시 상호 주소를 통하여 데이터를 주고받는 통신을 처리하게 됩니다. IP 주소는 컴퓨터들 간의 통신을 위하여 인식할 수 있는 특수한 번호입니다.

IP 주소는 0~255까지의 표현할 수 있는 1바이트의 수, 4개로 구성되어 있습니다. 또한 각각의 수 사이에는 점(.)을 통하여 구분하여 표기를 합니다.

xxx.xxx.xxx.xxx

즉 인터넷 IP 주소는 4개의 바이트인 32비트의 값으로 표현을 하게 됩니다. 32비트는 256×256×256×256 = 4,294,967,296개의 IP 주소를 생성할 수 있습니다.

이렇게 32비트로 구성된 IP 주소를 IPv4라고 합니다. 우리가 일반적으로 알고 있는 IP 주소 형식입니다.

IP 주소 관리 업무는 APNIC 등 국제기구에서 합니다. 또한 국가별로 IP 주소를 대행하여 관하는데, 대한민국의 경우 인터넷 주소 자원 관리 기관에서 업무를 처리하고 있습니다.

09.1.2 IP 주소의 구성

IP 주소는 4바이트로 구성된 32비트 값입니다. 전 세계 접속한 수많은 컴퓨터들을 구분하기 위해서 IP 주소는 크게 2개의 부분으로 나누어 IP 값을 분리합니다.

IP 주소 = 네트워크 주소 + 호스트 주소

이렇게 범위를 분리하는 이유는 큰 그룹의 네트워크와 회사 내 호스트 컴퓨터인지를 구분하기 위함입니다. IP 주소 구분을 명칭하는 다른 이름으로는 CLASS라는 표현을 사용합니다.

네트워크 주소

4개로 구성된 IP 주소의 첫 번째 0~255까지의 값을 네트워크 주소로 명칭합니다. 첫 숫자는 반드시 0에서 127 사이에 있어야 하는 것이 규칙입니다.

호스트 주소

나머지 3개의 숫자를 호스트 주소로 명칭합니다. 이 3개의 숫자값을 기준으로 CLASS A, B, C 형태로 구분하여 표시합니다. 두 번째 숫자의 그룹을 A 클래스, 세 번째 숫자를 B 클래스, 네 번째 숫자를 C 클래스라고 표현합니다.

클래스 A

A 클래스는 **숫자.xxx.xxx.xxx** 형태로 256×256×256개 = 16,777,216개의 호스트를 할당

하고 관리할 수 있습니다. A 클래스 주소는 많은 수의 컴퓨터를 할당하고 관리할 수 있습니다. 주로 큰 기관에서 할당되는 클래스입니다.

클래스 B

B 클래스는 **숫자.숫자.xxx.xxx** 형태로 256×256개 = 65,536개의 호스트를 할당하고 관리할 수 있습니다.

클래스 C

C 클래스는 **숫자.숫자.숫자.xxx** 형태로 256의 호스트를 할당하고 관리할 수 있습니다.

서브넷 마스크

서브넷 마스크(Subnet Mask)는 IP 주소를 비트 연산을 통하여 필터링해주는 특수한 값입니다. 또 다른 표현으로 네트워크 마스크(Network mask)라고도 합니다. 서브넷 마스크의 역할은 네트워크 부분과 호스트를 클래스 단위로 구분하는 것입니다.

```
Class A   255. 0. 0. 0
Class B   255. 255. 0. 0
Class C   255. 255. 255. 0
```

게이트웨이

게이트웨이(Gateway)란 네트워크의 특수한 IP 주소 입니다. 서브넷 마스크로 필터링된 클래스 안은 내부 네트워크입니다. 다른 외부의 네트워크와는 차단되어 있습니다. 하지만 인터넷과 같이 외부의 네트워크로 연결하기 위해서는 게이트웨이라는 특별한 IP 주소를 통하여 외부와 통신을 처리하게 됩니다.

게이트웨이는 자신의 속한 클래스 안에서 임의의 IP 주소를 하나 할당하여 사용하면 됩니다. 보통 게이트웨이 주소로는 xxx.xxx.xxx.1 또는 xxx.xxx.xxx.254와 같이 첫 번째, 마지막 번호를 자주 사용합니다.

맥 주소

맥 주소(Mac Address)는 IP 주소가 할당되는 하드웨어의 고유 기기 번호입니다. IP 주소는 네트워크에 연결된 컴퓨터 주소를 말하지만, 맥 주소는 컴퓨터 하드웨어 자체의 네트워크 장비의 번호입니다.

맥 주소는 6개의 바이트로 구성됩니다. 또한 헥사 코드값으로 표현하며, 각각의 값 사이에는 콜론(:)으로 구분합니다.

xx:xx:xx:xx:xx:xx

09.1.3 IP6V

기존 32비트 주소 체계인 IP4v 는 인터넷이 처음 시작되던 시점에는 전 세계의 모든 컴퓨터를 커버할 수 있을 것이라 생각했습니다. 하지만 전 세계 컴퓨터 사용량이 늘어나고, 모바일 기기의 등장으로 한 사람이 사용하는 디지털 장비들은 더욱 늘어나게 되었습니다.

그래서 우리는 편법으로 현재 사용하는 장비에만 IP 주소를 임시적으로 할당하는 편법을 사용하게 되었습니다. 동적 IP 주소(dynamic IP address)는 주소를 장비에 할당했다가 사용하지 않을 때는 회수하여 다른 장비에 할당하는 기술입니다.

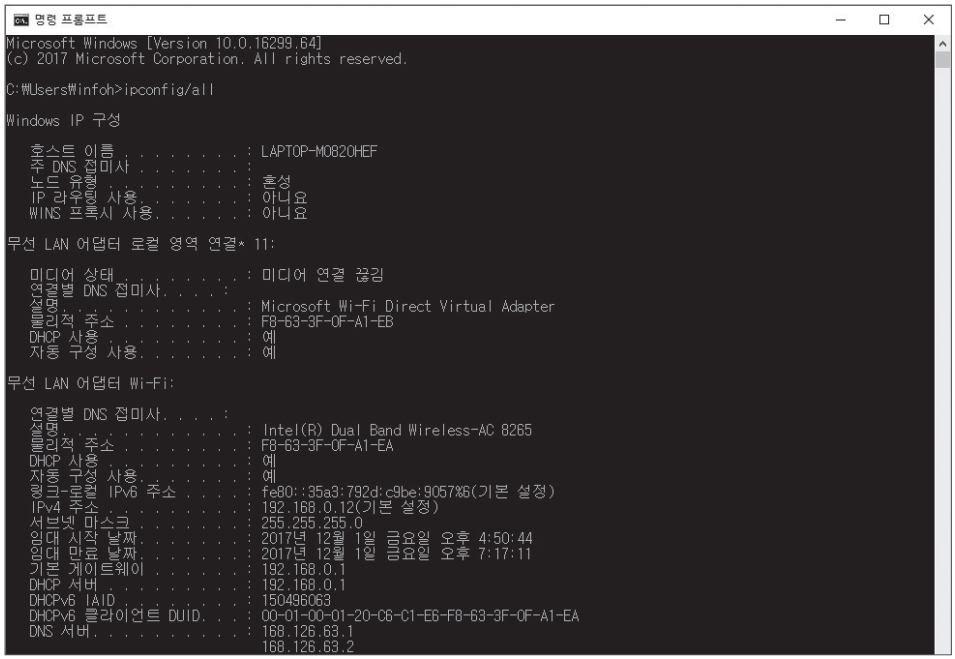
하지만 이러한 기술로도 더 이상 늘어나는 디지털 장비들을 수용하기란 어려워졌습니다. 그래서 등장한 새로운 IP 주소 체계가 IP6v입니다.

기존 IP4v의 42억9496만7296개의 숫자는 이제 충분한 숫자가 아니게 되었습니다. 이 부족한 숫자 체계를 보완하고자 만든 것이 IP6v 128비트 체계입니다.

09.1.4 IP 및 맥 주소 확인 방법

윈도우

콘솔 창에서 ipconfig/all 명령을 입력하면 본인 컴퓨터의 IP 주소와 맥주소를 확인할 수 있습니다.



리눅스

터미널 창에서 ipconfig 명령을 통하여 확인할 수 있습니다.

안드로이드

'설정' → '일반' → '휴대폰정보' → '하드웨어 정보'를 선택합니다.

09.1.5 도메인 주소

IP 주소를 통하여 서버 컴퓨터를 찾기란 매우 어렵습니다. 접속하려고 하는 모든 서버의 IP 주소를 외워서 인터넷을 사용하기란 힘듭니다. 따라서 이러한 IP 주소들을 가상의 별칭으로 할당하여 사용하는 것을 도메인이라고 합니다.

도메인은 복잡한 숫자 체계인 IP 주소보다 사람이 쉽게 이해할 수 있는 이름 형태로 되어 있어서 좀 더 직관적으로 인터넷을 접속하고 사용할 수 있습니다.

인터넷 브라우저 창에서 도메인 주소를 입력하면 해당 서비스 페이지로 접속합니다. 하지만 도메인 이름으로 어떻게 해당 서버를 찾을 수 있을까요? 이런 작업이 가능한 이유는 도메인 이름을 IP 주소로 변경 및 관리해주는 DNS 서버가 있기 때문입니다.

DNS 서버들은 서로 유기적으로 동작을 하면서 도메인 이름을 IP 주소로 변경하는 기능을 합니다. 서비스를 운영하는 서버들은 자체적으로 DNS 서비스를 운영하거나, 호스팅 회사에서 운영하는 DNS 서비스를 이용하기도 합니다.

예전에는 1개의 도메인 이름에 1개의 서버 IP를 할당하여 서비스를 구축했습니다. 하지만 사용자 접속이 많아지면서 1대의 서버만으로는 서비스를 운영하기는 매우 어려울 것입니다. 만일 www.google.com 사이트가 서버 1대로 전 세계의 사용자에게 서비스한다면 그 서버는 얼마나 큰 장비일까요? 대형 서비스를 운영하기 위해서는 다수의 서버 장비의 IP를 하나의 도메인으로 연결하여 사용자 접속을 분산합니다.

이런 점에서 IP 주소만 가지고 서비스를 만들거나 접속할 수는 없습니다. 도메인은 쉽게 IP 주소를 일반적인 이름으로 변경해주는 것과 동시에 대용량의 서비스를 구축할 수 있도록 환경을 제공합니다.

09.1.6 URL 쿼리스트링

도메인은 인터넷 서비스를 접속하는 데 매우 유용한 접속 방법입니다. 또한 웹 서비스들은 다양한 정보를 사용자에게 제공합니다.

사용자가 다양한 웹 서비스의 정보를 얻기 위해서 각각의 정보마다 도메인 URL을 제공한다고 하면 DNS 시스템은 과부하를 걸치게 될 것입니다. 따라서 웹 서비스들은 하나의 도메인에 각각의 서비스를 요청 정보를 함께 전송할 수 있는 쿼리스트링을 지원합니다.

쿼리스트링을 다른 표현으로 GET 방식의 URL 요청이라고 하기도 합니다. 쿼리스트링은 다음과 같이 표현합니다.

http://www.도메인.com?변수=값&변수=값&변수=값

도메인 주소 다음에 ? 기호는 쿼리스트링을 시작하는 표시입니다. 쿼리스크링은 '변수=값'과 같이 한 쌍의 데이터로 표현하며, 다수의 데이터를 전송할 때는 & 기호로 구분합니다.

쿼리스트링을 이용하면 쉽게 웹 서비스의 데이터를 선택하여 접속할 수 있습니다.

09.2 URL 함수

PHP에서는 URL 구조를 분석하고 처리할 수 있는 몇 개의 내장 함수들을 지원합니다. URL 함수를 이용하면 복잡한 URL 문자열을 하나씩 처리하지 않고 빠르게 작업을 수행할 수 있습니다.

09.2.1 URL 구분

내장 함수 parse_url()은 URL 스트링을 해석하여 구성요소를 배열로 반환합니다.

| 내장 함수 |

```
mixed parse_url ( string $url [, int $component = -1 ] )
```

예제 파일 | **url-01.php**

```
<?php
    $url = 'http://username:password@hostname:9090/path?arg=value#anchor';

    // url을 분석하여 배열로 반환합니다.
    print_r(parse_url($url));
    echo "<br>";

    // 키 값을 통해서 하나씩 추출이 가능합니다.
    echo "PHP_URL_SCHEME = ". parse_url($url, PHP_URL_SCHEME)."<br>";

    echo "PHP_URL_USER =". parse_url($url, PHP_URL_USER) ."<br>";
```

```
12      echo "PHP_URL_PASS =". parse_url($url, PHP_URL_PASS) ."<br>";
13      echo "PHP_URL_HOST =". parse_url($url, PHP_URL_HOST) ."<br>";
14      echo "PHP_URL_PORT =". parse_url($url, PHP_URL_PORT) ."<br>";
15      echo "PHP_URL_PATH =". parse_url($url, PHP_URL_PATH) ."<br>";
16      echo "PHP_URL_QUERY =". parse_url($url, PHP_URL_QUERY) ."<br>";
17      echo "PHP_URL_FRAGMENT =". parse_url($url, PHP_URL_FRAGMENT) ."<br>";
18
19  ?>
```

화면 출력

```
Array ( [scheme] => http [host] => hostname [port] => 9090 [user] => username
[pass] => password [path] => /path [query] => arg=value [fragment] => anchor )
PHP_URL_SCHEME = http
PHP_URL_USER =username
PHP_URL_PASS =password
PHP_URL_HOST =hostname
PHP_URL_PORT =9090
PHP_URL_PATH =/path
PHP_URL_QUERY =arg=value
PHP_URL_FRAGMENT =anchor
```

| 내장 함수 |

string **http_build_query** (mixed $query_data [, string $numeric_prefix [, string $arg_separator [, int $enc_type = PHP_QUERY_RFC1738]]])

내장 함수 http_build_query()는 URL 인코딩 쿼리 문자열을 생성합니다.

예제 파일 | **http_build_query.php**

```
1   <?php
2      $data = array(
3         'foo' => 'bar',
4         'baz' => 'boom',
5         'cow' => 'milk',
6         'php' => 'hypertext processor'
7      );
8
```

```
9       echo http_build_query($data) . "<br>";
10      echo http_build_query($data, '', '&amp;');
11
12  ?>
```

화면 출력

```
foo=bar&baz=boom&cow=milk&php=hypertext+processor
foo=bar&baz=boom&cow=milk&php=hypertext+processor
```

09.3 암호화

암호화는 URL의 쿼리스트링 작성 시 발생할 수 있는 오류나 다른 사용자에게 쉽게 url 값이 노출되는 것을 방지할 때 사용합니다. url 암호화는 인코딩 작업과 디코딩 작업의 한 쌍으로 동작합니다.

09.3.1 URL 인코딩

URL 주소는 불특정 다수의 사람들에게 오픈되어 있습니다. 오픈되어 있는 URL 주소는 쉽게 읽을 수 있습니다. PHP는 URL에 대해서 인코딩 작업을 할 수 있는 전용 함수를 제공합니다.

| 내장 함수 |

```
string urlencode ( string $str )
```

내장 함수 urlencode()는 주어진 url 문자열을 인코딩하여 반환합니다.

| 내장 함수 |

```
string urldecode ( string $str )
```

내장 함수 urldecode()는 인코딩된 url을 디코딩합니다.

예제 파일 | **url-02.php**

```php
<?php
    $url = "http://www.hojin.io/index.php";
    $encode = urlencode($url);
    echo "인코딩 = ". $encode."<br>";
    echo "디코드 = ". urldecode($encode)."<br>";

    $encode2 = urlencode("?aaa=1&bbb=2");
    echo "인코딩 = ". $url.$encode2."<br>";
    echo "디코드 = ". $url.urldecode($encode2)."<br>";

?>
```

화면 출력

```
인코딩 = http%3A%2F%2Fwww.hojin.io%2Findex.php
디코드 = http://www.hojin.io/index.php
인코딩 = http://www.hojin.io/index.php%3Faaa%3D1%26bbb%3D2
디코드 = http://www.hojin.io/index.php?aaa=1&bbb=2
```

09.3.2 base64 URL 처리

PHP는 base64 기반으로 URL을 인코딩, 디코딩하여 처리할 수 있습니다. base64와 관련된 전용 함수들을 제공합니다.

| 내장 함수 |

```
string base64_encode ( string $data )
```

내장 함수 base64_encode()는 base64 방식으로 인코딩합니다.

| 내장 함수 |

string **base64_decode** (string $data [, bool $strict = false])

내장 함수 base64_decode()는 base64 방식으로 인코딩합니다.

예제 파일 | **url-03.php**

```
1  <?php
2     $str = "안녕하세요! 지니PHP입니다.";
3     $encode = base64_encode($str);
4     echo "인코딩 = ". $encode ."<br>";
5
6     $decode = base64_decode($encode);
7     echo "디코딩 = ". $decode ."<br>";
8
9  ?>
```

화면 출력

```
인코딩 = 7JWI64WV7ZWY7IS47JqUISDsp4Dri4hQSFDsnoXri4jri6Qu
디코딩 = 안녕하세요! 지니PHP입니다.
```

09.3.3 RFC1738 URL 처리

PHP는 RFC1738 기반으로 URL을 인코딩, 디코딩하여 처리할 수 있습니다. RFC1738와 관련된 전용 함수들을 제공합니다.

| 내장 함수 |

string **rawurlencode** (string $str)

내장 함수 rawurlencode()는 RFC1738 형식에 따라서 URL을 인코딩합니다.

string **rawurldecode** (string $str)

내장 함수 rawurldecode()는 인코딩된 url 문자열을 디코딩 처리합니다.

예제 파일 | **string-27.php**

```
1  <?php
2     $url="http://www.hojin.io/?name=jiny&country=korea";
3     echo "URL = " . $url . "<br>";
4
5     $urlEncode = rawurlencode($url);
6     echo "인코딩 : ". $urlEncode . "<br>";
7
8     $urlDecode = rawurldecode($urlEncode);
9     echo "디코딩 : ". $urlDecode . "<br>";
10
11 ?>
```

화면 출력

```
URL = http://www.hojin.io/?name=jiny&country=korea
인코딩 : http%3A%2F%2Fwww.hojin.io%2F%3Fname%3Djiny%26country%3Dkorea
디코딩 : http://www.hojin.io/?name=jiny&country=korea
```

09.4 호스트 정보

PHP는 접속된 단말기의 정보들을 읽어서 처리할 수 있습니다. 시스템을 운영하면서 로그 기록, 통계 등을 위해서 클라이언트 단말기의 접속 정보 등을 처리할 수 있습니다.

09.4.1 IP 주소 확인

내장 함수 gethostbyname()은 지정한 인터넷 호스트 이름에 대하여 IPv4 주소 정보를 가져옵니다.

| 내장 함수 |

string **gethostbyname** (string $hostname)

예제 파일 | gethostbyname.php

```
1  <?php
2      $ip = gethostbyname('www.hojin.io');
3      echo $ip;
4
5  ?>
```

화면 출력 93.184.216.34

| 내장 함수 |

array **gethostbynamel** (string $hostname)

내장 함수 gethostbynamel()은 지정된 인터넷 호스트 이름에 해당하는 IPv4 주소 목록을 가져옵니다.

예제 파일 | gethostbynamel.php

```
1  <?php
2      $hosts = gethostbynamel('www.hojin.io');
3      print_r($hosts);
4
5  ?>
```

화면 출력

```
Array ( [0] => 93.184.216.34 )
```

09.4.2 호스트명

내장 함수 gethostbyaddr()은 ip_address로 지정된 인터넷 호스트의 호스트 이름을 반

환합니다.

| 내장 함수 |

string **gethostbyaddr** (string $ip_address)

예제 파일 | gethostbyaddr.php

```
<?php
    $hostname = gethostbyaddr($_SERVER['REMOTE_ADDR']);
    echo $hostname;

?>
```

| 내장 함수 |

string **gethostname** (void)

내장 함수 gethostname()은 호스트 이름을 가져옵니다.

예제 파일 | gethostname.php

```
<?php
    // 호스트 이름을 출력합니다.
    echo gethostname();

?>
```

화면 출력

```
ns.dojangshop.com
```

09.4.3 포트

내장 함수 getservbyname()은 인터넷 서비스 및 프로토콜과 관련된 포트 번호를 가져옵니다.

| 내장 함수 |

int **getservbyname** (string $service , string $protocol)

예제 파일 | **getservbyname.php**

```
<?php
    $services = array('http', 'ftp', 'ssh', 'telnet', 'imap',
        'smtp', 'nicname', 'gopher', 'finger', 'pop3', 'www');

    foreach ($services as $service) {
        $port = getservbyname($service, 'tcp');
        echo $service . ": " . $port . "<br />\n";
    }

?>
```

화면 출력

```
http: 80
ftp: 21
ssh: 22
telnet: 23
imap: 143
smtp: 25
nicname: 43
gopher: 70
finger: 79
pop3: 110
www: 80
```

09.5 DNS 정보

PHP는 DNS 정보를 확인할 수 있는 몇 개의 내장 함수를 지원합니다.

| 내장 함수 |

bool **checkdnsrr** (string $host [, string $type = "MX"])

내장 함수 checkdnsrr()은 주어진 인터넷 호스트 이름 또는 IP 주소에 해당하는 DNS 레코드를 확인하면 됩니다.

예제 파일 | **checkdnsrr.php**

```
<?php
    $email = "infohojin@naver.com";
    $exp = "^[a-z\'0-9]+([._-][a-z\'0-9]+)*@([a-z0-9]+([._-][a-z0-
    9]+))+$";

    if (eregi($exp,$email)) {
        $emailAddr = explode("@",$email);
        if (checkdnsrr($emailAddr[1],"MX")) {
            echo "DNS 레코드 확인";
        } else {
            echo "DNS 레코드 확인 불가";
        }

    } else {
      echo "이메일 형식이 아닙니다.";
    }

?>
```

화면 출력

DNS 레코드 확인

| 내장 함수 |

array **dns_get_record** (string $hostname [, int $type = DNS_ANY [, array &$authns [, array &$addtl [, bool $raw = false]]]])

내장 함수 dns_get_record()는 호스트 이름과 관련된 DNS 리소스 레코드를 가져옵니다.

예제 파일 | dns_get_record.php

```
<?php
    $result = dns_get_record("php.net");
    print_r($result);

?>
```

화면 출력

```
Array ( [0] => Array ( [host] => php.net [type] => AAAA [ipv6] => 2a02:cb41::7
[class] => IN [ttl] => 300 ) [1] => Array ( [host] => php.net [type] => A
[ip] => 72.52.91.14 [class] => IN [ttl] => 300 ) [2] => Array ( [host] =>
php.net [type] => TXT [txt] => v=spf1 ip4:72.52.91.12 ip6:2a02:cb41::8
ip4:140.211.15.143 ?all [entries] => Array ( [0] => v=spf1 ip4:72.52.91.12
ip6:2a02:cb41::8 ip4:140.211.15.143 ?all ) [class] => IN [ttl] => 300 ) [3] =>
Array ( [host] => php.net [type] => MX [pri] => 0 [target] => php-smtp2.php.
net [class] => IN [ttl] => 30 ) [4] => Array ( [host] => php.net [type] => SOA
[mname] => ns1.php.net [rname] => admin.easydns.com [serial] => 1484930803
[refresh] => 16384 [retry] => 2048 [expire] => 1048576 [minimum-ttl] => 2560
[class] => IN [ttl] => 300 ) [5] => Array ( [host] => php.net [type] => NS
[target] => dns2.easydns.net [class] => IN [ttl] => 300 ) [6] => Array ( [host]
=> php.net [type] => NS [target] => dns3.easydns.org [class] => IN [ttl] =>
300 ) [7] => Array ( [host] => php.net [type] => NS [target] => dns1.easydns.
com [class] => IN [ttl] => 300 ) [8] => Array ( [host] => php.net [type] => NS
[target] => dns4.easydns.info [class] => IN [ttl] => 300 ) )
```

| 내장 함수 |

bool **getmxrr** (string $hostname , array &$mxhosts [, array &$weight])

내장 함수 getmxrr()은 지정된 인터넷 호스트 이름에 해당하는 MX 레코드를 체크합니다.