Chapter

04

JSON

JSON 포맷은 더글라스 크록포드에 의해서 RFC7159와 ECMA-404라는 표준에 의해서 정의된 기술입니다. ECMA-404는 표준 문법만 제공하며, RFC7159는 보안적인 시맨틱 부분을 고려한 문법을 제공합니다.

JSON이란 Javascript Object Notation의 약자로 XML과 비슷한 데이터의 집합의 텍스트 데이터입니다. 주로 데이터 교환을 위하여 최근에 많이 사용하는 데이터 포맷의 일종입니다.

최근 들어 비동기 통신(AJAX)이 인기를 얻고 있는 가운데 간단한 데이터 값의 서로 전송을 위하여 JSON 규격의 문자열 데이터를 많이 사용합니다. JSON은 특히 인터넷에서 데이터를 주고받는데 더욱더 유용합니다. 대부분의 웹 API들은 JSON 방식의 데이터 처리와 교환을 위한 포맷으로 이용합니다.

JSON 데이터는 C, C++, PHP 등 프로그래머들에게 친숙한 텍스트 규격 형태로 초보 개발자들도 쉽게 이해할 수 있습니다. JSON 파일을 처리하기 위해서는 복잡한 문자열 처리 루틴이 필요합니다. 하지만 최신 버전의 PHP는 최신 트렌드의 JSON을 쉽게 처리할 수 있는 내장 함수를 제공합니다.

04.1 JSON 문법

JSON은 다양한 형식의 데이터를 문자열을 이용하여 데이터 직렬화 처리를 합니다. 여러 개의 데이터를 하나의 문자열로 표현하는 데 있어 약간의 규칙과 문법 구조를 가지고 있습니다.

04.1.1 문자열 표현

JSON에서 키와 데이터를 표현하는 문자열은 큰따옴표(")로 묶어서 작성합니다. JSON 데이터를 PHP 소스 내에서 사용할 때 큰따옴표는 기존 문자열과 함께 처리하는 데 충돌이 발생할 수 있습니다.

PHP에서 JSON 데이터 문자열을 삽입할 때 JSON 안에 있는 큰따옴표는 백슬래시(\")를 추가하여 사용합니다.

04.1.2 데이터 표현

JSON에서 데이터는 키/값 한 쌍의 데이터로 표시합니다. collection 타입 표기라고도 합니다. 이러한 표기는 object, struct(구조체), 연상 배열과 같은 문법 표현에서 자주 사용합니다.

| 표현 |

```
"키":"값"
```

키 이름은 변수와 같으며, 값은 그 변수의 데이터 값과 같다고 합니다. JSON은 배열처럼 여러 개의 키/값을 갖는 다수의 데이터로 구성할 수 있습니다. 즉 연상 배열처럼 키와 값 형태의 한 쌍으로 다수의 데이터를 처리할 수 있습니다.

04.1.3 배열 표현

JSON 데이터를 배열로 처리할 수 있습니다. 데이터는 키와 값 형태로 묶어서 입력합니다. 그리고 대괄호를 이용하여 감싸면 됩니다.

예제 파일 | **json-01.php**

```
1  <?php
2
3      $string = "
4          [10, {\"a\":20}, [30,\"서른\"] ]
5      ";
6
7      echo "=== JSON 문자열 ===<br>";
8      echo $string;
9      echo "<br>";
10
11     echo "=== 배열 처리 ===<br>";
12     $arr = json_decode($string);
13     print_r($arr);
14
15 ?>
```

화면 출력

```
=== JSON 문자열 ===
[10, {"a":20}, [30,"서른"] ]
=== 배열 처리 ===
Array ( [0] => 10 [1] => stdClass Object ( [a] => 20 ) [2] => Array ( [0] => 30 [1] => 서른 ) )
```

04.1.4 객체 표현

JSON의 데이터를 객체로 표현할 수 있습니다. 각각의 데이터를 중괄호로 표기하면 됩니다.

| 표현 |

```
{
      "키":"값",
       "키":"값",
       "키":"값",
       "키":"값"
}
```

예제 파일 | **json-02.php**

```
1   <?php
2
3       $string = "
4           {\"name1\":10,\"name2\":\"안녕하세요\",\"name3\":true}
5       ";
6
7       echo "=== JSON 문자열 ===<br>";
8       echo $string;
9       echo "<br>";
10
11      echo "=== 배열 처리 ===<br>";
12      $arr = json_decode($string);
13      print_r($arr);
14
15  ?>
```

화면 출력

```
=== JSON 문자열 ===
{"name1":10,"name2":"안녕하세요","name3":true}
=== 배열 처리 ===
stdClass Object ( [name1] => 10 [name2] => 안녕하세요 [name3] => 1 )
```

JOSN 문자열은 비순서화된 데이터 세트입니다. 중괄호 안에 여러 개의 데이터를 넣을 수 있기 때문입니다. 여러 개의 데이터를 묶을 때는 각각의 데이터를 콤마(,)로 구분하고 전체를 중괄호로 감싸면 됩니다.

04.1.5 객체 이중화 표현

하나의 그룹으로 만들어진 JSON 데이터를 이중으로 다시 묶을 수도 있습니다. JSON 객체 안에 또 다른 JSON 객체를 갖는 형태입니다. 이중 배열처럼 다중 관계를 가지는 구조라고 이해할 수 있습니다.

| 표현 |

```
{
      {
            “키”:”값”,
            “키”:”값”,
            “키”:”값”,
            “키”:”값”
      },
      {
            “키”:”값”,
            “키”:”값”,
            “키”:”값”,
            “키”:”값”
      }
}
```

이때 또한 각각의 JSON 데이터를 콤마(,)로 구분하고 다시 전체를 중괄호를 이용하여 감싸 묶을 수 있습니다.

04.1.6 객체 키 표현

JSON은 중괄호를 하나의 object, array로 인식합니다. 또한 각각의 데이터는 콤마를 이용하여 구분합니다. 또한 콜론(:) 앞은 키 이름으로 인식합니다.

이중 배열처럼 JSON 데이터의 키 값이 있을 경우에는 “키”:{} 형태로 지정할 수 있습니다.

| 표현 |

```
{
        "1":{
                "FirstName":"lee",
                "LastName":"hojin"
        }
}
```

여러 개의 이중 배열 형태로 묶기 위해서는 대괄호를 사용할 수도 있습니다.

쌍따옴표(")로 둘러싸인 키와 쌍따옴표(")로 둘러싸인 값을 콜론(:) 기호로 연결합니다. 또한 여러 개의 데이터는 콤마(,)로 구분합니다.

04.1.7 주석

JSON 데이터는 띄어쓰기나 탭을 통하여 보기 좋게 코드를 정리할 수는 있으나 주석과 같은 설명문은 넣을 수가 없습니다.

04.2 JSON 인코딩

소스상의 데이터, 배열을 JSON 형태의 문자열로 변환할 수 있습니다. 인코딩은 디코딩의 반대말입니다. PHP는 JSON 인코딩을 쉽게 처리하기 위해서 전용 함수를 제공합니다.

| 내장 함수 |

```
string json_encode ( mixed $value [, int $options = 0 [, int $depth = 512 ]] )
```

내장 함수 json_encode()를 통하여 간단하게 배열 데이터를 JSON 형태의 문자열로 직렬화 변환을 할 수 있습니다.

예제 파일 | **json-03.php**

```php
<?php

    $arr['FirstName'] = "lee";
    $arr['LastName'] = "hojin";

    $json = json_encode($arr);
    echo $json;

?>
```

화면 출력

```
{"FirstName":"lee","LastName":"hojin"}
```

위의 실험을 보면 배열 변수를 인코딩 함수를 이용하여 JSON 문자열로 변경합니다. json으로 직렬화된 문자열을 출력합니다.

예제 파일 | **json-04.php**

```php
<?php

    $arr['FirstName'] = "lee";
    $arr['LastName'] = "hojin";
    $user['1'] = $arr;

    $arr['FirstName'] = "jiny";
    $arr['LastName'] = "PHP";
    $user['2'] = $arr;

    $json = json_encode($user);
    echo $json;

?>
```

화면 출력

```
{
"1":{
"FirstName":"lee",
"LastName":"hojin"
```

```
},
"2":{
"FirstName":"jiny",
"LastName":"PHP"
}
}
```

위의 실험은 다차원 배열의 데이터를 JSON으로 인코딩하는 예제입니다. 배열에서 연상 키 값이 있으면 "키":{} 형태로 변경되는 것을 확인할 수 있습니다. 또한 다차원 배열을 JSON 직렬화하면서 중괄호를 계속 중첩하여 사용합니다. 다차원 적인 배열도 JSON 데이터 직렬화를 처리할 수 있습니다.

04.3 JSON 디코딩

직렬화된 JSON은 데이터를 담고 있는 텍스트 문자열과 같습니다. 데이터를 직렬화하여 처리하는 것은 이기종 간에 데이터를 전달할 때 매우 편리합니다. 직렬화 변환된 데이터를 저장하고 다른 시스템으로 전송도 가능합니다.

04.3.1 배열 디코딩

내장 함수 json_decode()를 통하여 간단하게 직렬화된 JSON 문자열을 데이터로 변환할 수 있습니다. 디코딩은 인코딩의 반대말입니다.

PHP는 JSON 파일을 PHP에서 사용할 수 있는 데이터 형태로 쉽게 변할 수 있도록 전용 함수를 제공합니다.

| 내장 함수 |

mixed **json_decode** (string $json [, bool $assoc = false [, int $depth = 512 [, int $options = 0]]])

예제 파일 | **json-05.php**

```php
<?php

    // 큰따옴표를 사용하기 위해서 백슬래시를 추가
    $string = "
    {
       \"Id\":\"01\",
       \"FirstName\": \"lee\",
       \"LastName\": \"hojin\",
       \"Country\": \"Korea\"
    }
    ";

    echo "=== JSON 문자열 ===<br>";
    echo $string;
    echo "<br>";

    echo "=== 배열 처리 ===<br>";
    $arr = json_decode($string);

    while (list($key,$val) = each($arr)) {
       echo $key. "==>" .$val. "<br>";
    }

?>
```

화면 출력

```
=== JSON 문자열 ===
{ "Id":"01", "FirstName": "lee", "LastName": "hojin", "Country": "Korea" }

=== 배열 처리 ===
Id==>01
FirstName==>lee
LastName==>hojin
Country==>Korea
```

04.3.2 다중 배열 디코딩

JSON은 1차원 데이터 이외에 다단계의 값을 갖는 다차원 배열 형태로 값을 지정할 수 있습니다.

다차원 배열을 지정하기 위해서는 중괄호 외에 대괄호를 사용합니다.

| 표현 |

```
[
      {
            “키”:”값”,
            “키”:”값”,
            “키”:”값”,
            “키”:”값”
      },
      {
            “키”:”값”,
            “키”:”값”,
            “키”:”값”,
            “키”:”값”
      }
]
```

대괄호는 배열 안에 또 다른 json 배열이 있다는 표기입니다.

예제 파일 | **json-06.php**

```
1  <?php
2
3     // 큰따옴표를 사용하기 위해서 백슬래시를 추가
4     $string = "
5
6        [
7           {
8              \"Id\":\"01\",
9              \"FirstName\": \"lee\",
```

```
            \"LastName\": \"hojin\",
            \"Country\": \"Korea\"
        },
        {
            \"Id\":\"02\",
            \"FirstName\": \"jiny\",
            \"LastName\": \"PHP\",
            \"Country\": \"Korea\"
        }
    ]

";

echo "=== JSON 문자열 ===<br>";
echo $string;
echo "<br>";

echo "=== 배열 처리 ===<br>";
$arr = json_decode($string);

for ($i=0;$i<count($arr);$i++) {
    echo "첫 번째 배열 $i <br>";

    while (list($key,$val) = each($arr[$i])) {
        echo $key. "==>" .$val. "<br>";
    }
}

?>
```

화면 출력

```
=== JSON 문자열 ===
[
{ "Id":"01", "FirstName": "lee", "LastName": "hojin", "Country": "Korea" },
{ "Id":"02","FirstName": "jiny", "LastName": "PHP", "Country": "Korea" }
]
=== 배열 처리 ===
첫 번째 배열 0
Id==>01
FirstName==>lee
```

```
LastName==>hojin
Country==>Korea
첫 번째 배열 1
Id==>02
FirstName==>jiny
LastName==>PHP
Country==>Korea
```

JSON 데이터를 소스상에서 직접 생성하여 처리할 수도 있지만, 외부 API 서비스를 접속하여 데이터를 수신한 후에 디코딩하여 프로그램에서 적용할 수 있습니다.

04.4 객체 직렬화

JSON 방식 이외에도 PHP 내에서 제공되는 함수를 통하여 객체를 직렬화 처리할 수 있습니다. 변환된 문자열은 파일 등으로 출력하여 저장할 수도 있고, 저장된 파일을 읽어서 객체로 다시 복원할 수도 있습니다.

| 내장 함수 |

string **serialize** (mixed $value)

내장 함수 serialize()는 저장 가능한 값에 대해서 바이트 스트림 표현의 문자열로 변경합니다. 객체 등을 변환하여 저장하거나 전달할 때 매우 유용합니다.

예제 파일 | **serialize.php**

```
1  <?php
2
3      class A {
4          public $one = 1;
5
6          public function show_one() {
7              echo $this->one;
8          }
```

```
    }

    $a = new A;

    // 클래스 인스턴스 객체를 직열화하여 저장합니다.
    $serialObj = serialize($a);
    file_put_contents('store', $serialObj);

    echo $serialObj;

?>
```

화면 출력

```
O:1:"A":1:{s:3:"one";i:1;}
```

| 내장 함수 |

mixed **unserialize** (string $str [, array $options])

내장 함수 unserialize()는 serialize() 함수를 통하여 직렬화된 기능을 역직렬화 작업을 수행합니다. 역직렬화를 하기 위해서는 직렬화된 문자열만으로는 처리할 수 없습니다. 역직렬화를 위한 이전의 클래스 객체의 정의가 함께 필요합니다.

예제 파일 | unserialize.php

```
<?php

    class A {
        public $one = 1;

        public function show_one() {
            echo $this->one;
        }
    }

    // 저장된 직열화 문자열 파일을 읽어서, 역직열화를 수행합니다.
    $s = file_get_contents('store');
```

```
13      $a = unserialize($s);
14
15      // $a 객체의 show_one() 함수를 호출합니다.
16      $a->show_one();
17
18  ?>
```

화면 출력

```
1
```