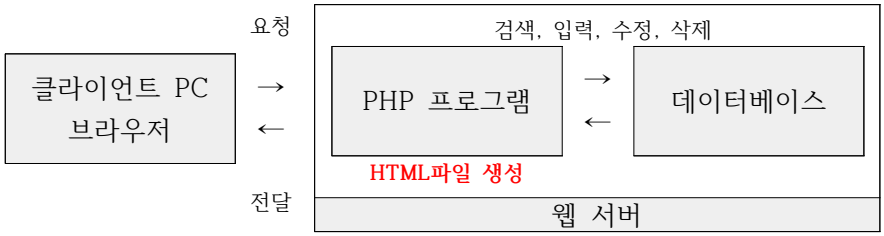
PHP(PHP – Hypertext Preprocessor)의 정의

- PHP는 Personal Home Page Tools라는 약자로 시작하여 현재는 PHP-Hypertext Preprocessor라는 약자로 통칭되며 서버 측에서 실행되는 서버 사이드(Server-side) 언어이다. PHP는 데이터베이스와 연동하여 데이터를 저장, 검색, 수정, 삭제할 수 있다. PHP는 무료로 공개되어 있으며 대부분의 서버에서 이용할 수 있다. 현재 대부분의 게시판 시스템이 PHP로 만들어져 있다. 본지에서는 5.6버전을 기준하였다.
- 클라이언트 컴퓨터에서 작동되는 javascript와는 달리 PHP는 서버 측 언어이기 때문에 클라이언트 컴퓨터에서 실행할 수 없다. 웹서버인 Apache, 스크립트 언어인 PHP, 데이터베이스로 사용할 mySQL이 모두 이용 가능한 서버에서만 동작이 가능하다. 대부분의 호스팅 업체에서는 이 세 가지 서비스가 가능하고 개인 컴퓨터에서 apmsetup 등의 가상 서버 프로그램을 이용하여 동작시킬 수 있다. 가상 서버에 대해서는 아래 링크를 참조하여 구축할 수 있다.
- https://www.google.com/search?q=apmsetup
- PHP는 웹을 위한 언어이며 웹 개발에 필요한 대부분의 기능들이 함수들로 준비되어 있다.



- 위 그림에서 설명하였듯 PHP 프로그램이 데이터베이스의 내용을 바탕으로 새로운 HTML파일을 생성하고 그렇게 생성된 HTML파일을 클라이언트가 받아보는 방식이므로 PHP는 서버 측 언어이며 서버에서만 작동한다.
- 문서의 확장자는 .php이나 .html로 작성해도 작동한다.

PHP 기본 문법

- HTML문서 내에서 <?php로 시작하여 ?>로 끝나는 문법을 통해 HTML 문서 안에서 PHP 모드 실행 영역을 설정할 수 있다.

```
<p>Hello World</p>
<?php
   // PHP Code
?>
<p>Hello World</p>
```

PHP 모드!

- **변수 명 앞에 반드시 $를 붙인다.** 변수를 선언할 때 별도의 명령어가 필요하지 않다. 넣는 값에 따라 정수형, 실수형, 문자열, 논리형(불리언), 배열형, 객체형 등의 형식이 정해진다. 변수 이름은 숫자로 시작할 수 없으며 알파벳으로 시작해야 한다. 알파벳과 숫자 _(언더바)를 사용할 수 있다.
- 문자열의 결합은 .(dot)을 사용하거나 {}(중괄호)를 사용하여 붙인다. (자바스크립트에서와 같이 +(더하기)를 사용하면 전혀 다른 결과를 나타낸다.)
- **문자열 내에서 변수명을 사용하면 해당 변수의 내용이 자동으로 삽입된다.**

```php
<?php
   $a = 4;
   $b = 5;
   $c = $a + $b;
   echo $a;    // 4를 출력함
   echo $c;    // 9를 출력함

   $firstName = "Jiho";
   $lastName = "Han";
   echo "My name is ".$firstName.$lastName;    // "My name is JihoHan"을 출력함
   echo "My name is $firstName $lastName";     // "My name is Jiho Han"을 출력함
   echo "My name is {$firstName} {$lastName}"; // "My name is Jiho Han"을 출력함
?>
```

- 기타 연산자, if문, for문 등의 문법은 javascript와 대부분 동일하다.

PHP 변수의 범위(scope)

- PHP 변수는 local, global, static 세 가지 중 하나의 범위를 가진다.
- local 변수는 함수의 내부에서 생성되고 함수가 끝날 때 지워지는 변수이다. 따라서 함수의 외부에서는 사용할 수 없다.

```php
<?php
   function apple(){
       $a = 4;                                  // 이 변수는 local 변수임
       echo "함수 내에서의 변수값 : $a";
   }
   apple();                                     // "함수 내에서의 local 변수값 : 4"를 출력함
   echo "함수 밖에서의 변수값 : $a";                    // "함수 내에서의 local 변수값 : "를 출력함
?>
```

- global 변수는 함수의 외부에서 생성되고 다른 함수의 내부에서는 사용할 수 없다.

```php
<?php
   $a = 4;                                      // 이 변수는 global 변수임
   function apple(){
       echo "함수 내에서의 변수값 : $a";
   }
   apple();                                     // "함수 내에서의 local 변수값 : "를 출력함
   echo "함수 밖에서의 변수값 : $a";                    // "함수 내에서의 local 변수값 : 4"를 출력함
?>
```

- global 키워드를 이용하여 함수 내부에서 함수 외부의 변수를 사용할 수 있다.

```php
<?php
   $a = 4;                                      // 이 변수는 global 변수임
   $b = 6;                                      // 이 변수는 global 변수임
   function apple(){
       global $a, $b;                          // 함수 내에서 global 변수를 불러옴
       $b = $a + $b;
   }
   apple();
   echo $b;                                     // 10을 출력함
?>
```

- $GLOBALS[*index*] 배열을 사용하여 global 변수를 불러올 수도 있다.

```php
<?php
   $a = 4;                                      // 이 변수는 global 변수임
   $b = 6;                                      // 이 변수는 global 변수임
   function apple(){
       $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
   }
   apple();
   echo $b;                                     // 10을 출력함
?>
```

자주 사용하는 PHP 키워드/ 변수

키워드	예시		설명
echo	echo "Hello World"; echo " Hello World "; echo "<script>alert('Hello World');</script>";		문서에 텍스트를 기입한다. html, css, script 모두 작동이 가능하다.
include	include "header.php";		해당 위치에 외부 파일을 로드한다.
exit	exit;		아래에 있는 코드들을 실행하지 않고 전체 프로그램을 끝낸다.
$GLOBALS	$x = 6; function apple(){ echo $GLOBALS['x']; // 6 }		함수 내부에서 전역변수를 로드한다.
$_POST	a.php	<form action="b.php" method="post"> <input name="no" value="2" /> </form>	post방식으로 불러들인 데이터를 로드한다.
	b.php	echo $_POST['no']; // 2	
$_GET	a.php	b.php?no=2	get방식으로 불러들인 쿼리의 데이터를 로드한다.
	b.php	echo $_GET['no']; // 2	
$_SESSION	a.php	session_start(); $_SESSION['no'] = 3;	서버와 클라이언트가 접속된 시점부터 서버에 정보를 기록한다. 페이지가 달라져도 세션기록은 계속 남아있다.
	b.php	echo $_SESSION['no']; // 3	

자주 사용하는 PHP 키워드/ 변수

키워드	예시		설명
$_FILES	a.php	<form action="b.php" method="post" enctype="multipart/form-data"> <input type="file" name="userfile" /> // abc.jpg를 로드함 </form>	업로드한 파일에 대한 설정값을 불러온다. 파일을 업로드하기 위해서는 **method는 반드시 post로 설정하며** **enctype="multipart/form-data"가** **선언되어 있어야 한다.** 해당 파일업로드의 파일명, 종류, 크기, 임시저장소이름을 추출할수 있다.
	b.php	echo $_FILES['userfile']['name']; // abc.jpg echo $_FILES['userfile']['type']; // image/jpeg echo $_FILES['userfile']['size']; // 56619 (byte단위) echo $_FILES['userfile']['temp_name']; // /tmp/php5Wx0aJ	

자주 사용하는 PHP 함수

함수	예시	설명
include(*파일명*)	include("header.php");	외부 파일을 가져와 그 내용을 현재 위치에 기입한다.
array(*배열값1,배열값2,...*)	$fruits[0] = "apple"; $fruits[1] = "banana"; $fruits[2] = "cherry"; $fruits = array("apple","banana","cherry"); $jiho['height'] = 175; $jiho['gender'] = "male"; $jiho = array("height"=>175,"gender"=>"male");	배열변수를 만든다. 객체형 변수처럼 값마다 이름을 정해줄 수 있다.
array_push(*배열,값*)	array_push($fruits,"ddalgi");	배열에 새 값을 추가한다.
count(*배열*)	count($jiho); // 결과는 2	배열의 데이터 개수를 센다.
echo(문자열)	echo("abcde"); // 결과는 "abcde"	문자열을 출력한다.
strlen(*문자열*)	strlen("abcde"); // 결과는 5	문자열의 글자 개수를 센다.
str_word_count(*문자열*)	str_word_count("abc de"); // 결과는 2	문자열의 단어 개수를 센다.
strpos(*문자열,패턴*)	strpos("abcde","cd"); // 결과는 2	문자열의 내용 중 특정 문자의 위치를 찾는다.
str_replace (*찾을패턴,새문자열,문자열*)	str_replace("cd","뿰뿰","abcde");// 결과는 ab뿰뿰e	문자열의 내용 중 특정 문자의 내용을 바꾼다.
explode(*찾을패턴,문자열*)	$a = explode(" ","Hello code world"); echo $a[1]; // 결과는 "code"	문자열의 내용 중 특정 패턴으로 문자열을 분해하여 배열로 저장한다.
implode(*추가패턴,문자열*)	$a = array("apple","banana","cherry"); echo implode(" ", $a); // 결과는 "apple banana cherry"	배열의 내용을 추가패턴을 중간에 추가하며 문자열로 저장한다.
nl2br(*문자열*)	nl2br("나랏말싸미\n듕귁에달아"); // 결과는 "나랏말싸미 듕귁에달아"	database의 데이터에서 줄바꿈을 하기 위해 쓴 이스케이프 문자(\n)를 로 바꿔준다.
time()	time();	1970년 1월 1일 0시 0분 0초 로부터 현재까지 지나온 시간(timestamp)을 ms단위로 반환한다.
date(*형식*)	date("Y-m-d"); // 2019-05-07	현재의 시간을 반환한다. Y(4자리 연도), y(2자리 연도), m(2자리 월), d(2자리 일)
move_uploaded_file (*임시저장소위치, 파일명*)	move_uploaded_file ($_FILES['*myfile*']['temp_name'], "upload/".$newfilename);	임시저장소에 업로드된 파일을 해당 파일명으로 저장한다.
unlink(*파일명*)	unlink("upload/".$filename);	해당 파일을 삭제한다.
isset(*변수명*)	$_SESSION['apple'] = "사과"; isset($_SESSION['apple']); // true	변수가 존재하는지 확인하여 참, 거짓을 반환한다.
abs(*숫자값*)	abs(-3.3); // 3.33	절대값을 구한다.
round(*숫자값*)	round(3.3); // 3	반올림값을 구한다.
ceil(*숫자값*)	ceil(3.3); // 4	올림값을 구한다.
floor(*숫자값*)	floor(3.3); // 3	버림값을 구한다.
pow(*숫자값,제곱수*)	pow(2, 4); // 2^4 즉 16	제곱승 값을 구한다.

1. 모듈 파일 만들기

1. common.php (공통속성 페이지)

mySQL로 접속을 하기 위한 코드, 매 페이지마다 로딩해야 할 사항들, 기타 설정 값 등을 작성한다. 이 파일은 현재 사이트 내의 대부분의 페이지에 연결될 파일로써 다른 페이지 최상단마다 include될 파일이다.

```php
<?php
header('Content-Type:text/html; charset=UTF-8;');                    // 헤더에 UTF-8 설정을 추가
// mysql 서버에 접속하기. localhost는 자기 자신 즉 서버를 지칭하는 이름이다.
$conn = mysqli_connect('localhost', 'DB아이디', 'DB비밀번호') or die("서버 접속 에러");
// DB 선택하기. 일반적으로 DB아이디로 DB명이 결정되는 경우가 많다.(호스팅 업체에 문의)
$result = mysqli_select_db('DB이름', $conn) or die("DB 선택 에러");
// mysql 서버에서 사용할 문자열 인코딩 설정을 utf-8로 지정한다.
mysqli_query($conn,"SET SESSION CHARACTER_SET_CONNECTION=utf8");
mysqli_query($conn,"SET SESSION CHARACTER_SET_RESULTS=utf8");
mysqli_query($conn,"SET SESSION CHARACTER_SET_CLIENT=utf8");
session_start();            // 세션 시작(세션을 사용하기 위한 세션기능 켜기)
// 사용자 헤더에 UTF-8 설정을 추가로그인한 사용자의 userlv(계정권한) 세션이 만들어져 있지 않으면(로그인 하지 않았다면) 그 값을 0으로 지정한다. 로그인 했다면 그 계정의 설정에 따라 userlv이 지정될 것이다.
if(!isset($_SESSION['userlv'])){
    $_SESSION['userlv'] = 0;
}
include "config.php"; // 게시판 설정 값
?>
```

2. header.php (상단 페이지)

html페이지의 헤드, 헤더 및 로고, 메뉴바 등을 작성한다. section, body, html 태그는 footer.php에서 닫힐 예정이므로 header.php에서 닫지 않는다.

```php
<!doctype html>
<html lang="ko">
<head>
    <meta charset="UTF-8"/>
    <meta name="viewport" content="initial-scale=1.0, width=device-width" />
    <title>리스트형 게시판</title>
    <link href="css/common.css" rel="stylesheet"/>
    <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
</head>
<body>
    <header>
        <h1>대제목</h1>
        <nav>메뉴바</nav>
        <?php
            if($_SESSION['userlv'] != 0){                    // 사용자 레벨이 0이 아니라면(로그인사용자라면)
                echo"<a href='logout.php'>로그아웃</a>";     // 로그아웃 버튼을 보여주고
            }else{
                echo"<a href='login.php'>로그인</a>";        // 사용자 레벨이 0이라면(비로그인사용자라면)
                echo"<a href='join.php'>회원가입</a>";       // 로그인, 회원가입 버튼을 보여준다
            }
        ?>
    </header>
    <section>
```

3. footer.php (하단 페이지)

html페이지의 하단에 표시될 푸터 및 스크립트 등을 작성한다. header.php에서 열었던 section, body, html 태그를 순서대로 닫는다.

```php
    </section>
    <script src="script/script.js"></script>
</body>
</html>
```

4. index.php (첫번째 페이지)

첫 번째 페이지인 index.php의 상단에 common.php, header.php 그리고 하단에 footer.php를 연결하여 완성된 html 문서를 작성한다.
기타 나머지 페이지들도 이와 같이 common, 헤더, 푸터를 연결하여 페이지들을 제작하게 된다.
헤더 및 푸터, 내용을 표시할 필요 없이 PHP상에서 데이터만 다루는 페이지의 경우 header.php와 footer.php를 연결할 필요가 없는 경우도 있다. 그러나 대부분의 페이지에서 common.php는 반드시 include하는 것이 일반적인 상황일 것이다.
※ common에서 작성된 내용은 내부적으로 작동하는 코드만 작성되어 있고 HTML로 출력하는 내용이 없으므로 표시되지 않음.

원본 PHP파일 ⇒ **클라이언트가 수신하게 될 HTML파일**

```php
<?php
include "common.php";
include "header.php";
?>

/* 본문내용 */

<?php
include "footer.php"
?>
```

⇒

```html
<!doctype html>
<html lang="ko">
<head>
    <meta charset="UTF-8"/>
    <meta name="viewport" content="initial-scale=1.0, width=device-width" />
    <title>리스트형 게시판</title>
    <link href="css/common.css" rel="stylesheet"/>
    <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
</head>
<body>
    <header>
        <h1>대제목</h1>
        <nav>메뉴바</nav>
        <?php
            if($_SESSION['userlv'] != 0){
                echo"<a href='logout.php'>로그아웃</a>";
            }else{
                echo"<a href='login.php'>로그인</a>";
                echo"<a href='join.php'>회원가입</a>";
            }
        ?>
    </header>
    <section>

/* 본문내용 */

    </section>
    <script src="script/script.js"></script>
</body>
</html>
```

5. config.php (게시판 파일)

각 게시판에서 사용될 각종 설정 값들을 미리 설정한다. 게시판 테이블의 이름, 읽기/쓰기 권한 설정, 한 페이지에서 보여줄 게시물 수, 각 페이지 블록별 보여줄 페이지 수, 세션 설정 등을 작성한다. 모든 페이지에서 적용되어야 하므로 common.php에 include한다.

config.php

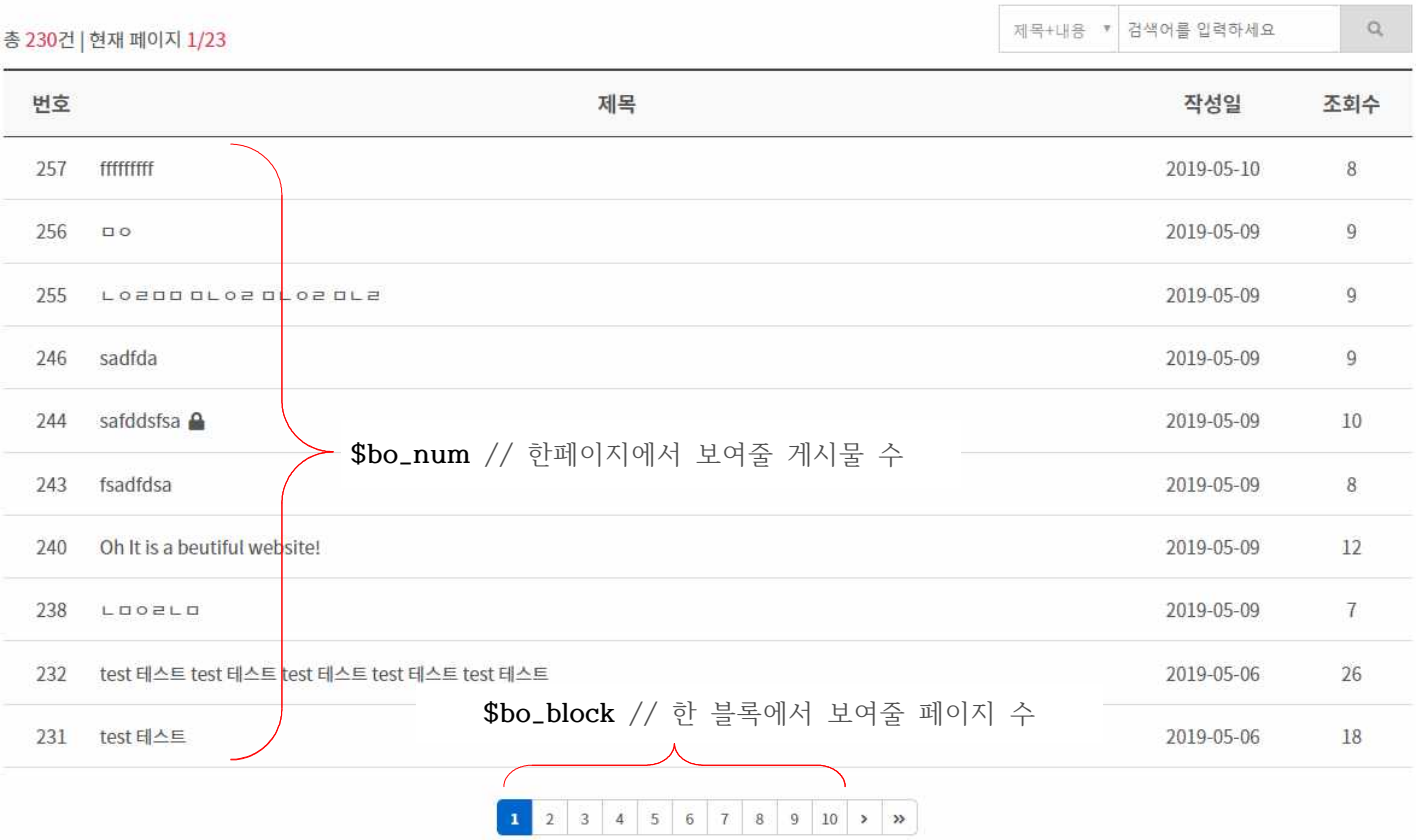
```
<?php
$notice_list = 0;          // notice게시판에서 비로그인 회원도 리스트 볼 수 있음
$notice_read = 0;          // notice게시판에서 비로그인 회원도 읽을 수 있음
$notice_write = 1;         // notice게시판에서 로그인 회원만 쓸수 있음

$qna_list = 0;             // qna게시판에서 비로그인 회원도 리스트 볼 수 있음
$qna_read = 0;             // qna게시판에서 비로그인 회원도 읽을 수 있음
$qna_write = 0;            // qna게시판에서 로그인 회원만 쓸수 있음

                           // ...

$bo_admin = 9;             // userlv9 이상은 관리자임
$bo_num = 10;              // 한 페이지에서 10개의 게시물을 표시함
$bo_block = 10;            // 한번에 보여줄 페이지 목록 수
?>
```

- 게시판 별 게시물 리스트 보기 권한
- 게시판 별 게시물 읽기 권한
- 게시판 별 게시물 쓰기 권한
- 게시판 관리자 권한
- 리스트 한 페이지에서 보여줄 게시물 수
- 한번에 보여줄 페이지목록 수



2. 사용자 권한 설정(회원가입/로그인)

1. join.php (회원가입 페이지)

1.1 회원가입

회원가입을 하는 페이지이다. 회원의 아이디, 비밀번호, 성명 등을 입력한다.
본 예제에서는 DB명을 [vision], 게시판의 테이블명을 [member]라고 명명하였다. [member]테이블의 스키마는 아래와 같다.
사용자 비밀번호는 hash함수로 암호화하여 저장하는데 이때 암호화된 텍스트의 길이가 대폭 늘어난다. 따라서 비밀번호 속성의 글자 수는 넉넉하게 만들어야 한다. 사용자 레벨은 게시판에서 읽기, 글쓰기, 삭제하기, 수정하기 등의 기능에서 사용 권한을 지정하기 위한 수치이며 0에서 9까지의 숫자를 사용하기로 하였으므로 속성의 길이는 1이다. 비로그인 사용자의 레벨은 0이고 로그인한 일반 회원은 1이며 최고 관리자는 9로 설정하기로 하였다.

[member]

속성명	no	userid	userpw	username	userlv
종류	INT	VARCHAR	TEXT	VARCHAR	INT
길이	10	200	300	100	1
데이터 정렬방식	-	UTF8UNICODE	UTF8UNICODE	UTF8UNICODE	-
설명	PRIMARY값으로서 각 게시물을 특정하는 유일값	사용자 아이디	사용자 비밀번호 (비밀번호가 hash로 암호화 되면 길이가 늘어나므로 300자 정도가 적당함)	사용자 이름	사용자 레벨

join.php

```php
<?php
include "common.php";
include "header.php";
?>
<form name="join" action="join_insert.php" method="post">
 <input type="text" name="userid" required />
 <input type="button" name="chkbtn" value="중복체크" onclick="dupchk();" />

 <input name="chkbool" type="hidden" value="0" />

 <iframe id="ifrm1" name="ifrm1" hidden></iframe>


 <input type="password" name="userpw" required />
 <input type="text" name="username" required />
 <button type="submit" onclick="chkval();return false;">가입하기</button>
</form>
<script>
   function dupchk() {
      var id = document.join.userid.value;
      document.join.chkbool.value = 0;
      ifrm1.location.href = 'idchk.php?userid='+id;
   }
   function chkval() {
      if(chkbool == 0) {
         alert("ID 중복체크를 해주세요.");
         return;
      }else{
         document.join.submit();
      }
   }
<script>
<?php
include "footer.php"
?>
```

설명

- join_insert.php로 데이터를 보낼 때 비밀번호가 있으므로 post방식으로 전송한다.
- 아이디 중복체크를 위해 dupchk()함수를 실행한다.
- 중복체크 후 중복이 없으면 chkbool의 값을 1로 바꾼다.
- 중복체크 시 member 테이블에서 아이디를 검색해야 하는데 지금 페이지를 벗어날 수 없으므로 숨겨진 iframe에서 idchk.php를 띄워 이를 진행할 것이다.

- 사용자가 입력한 아이디를 member 테이블에서 검색하기 위해 idchk.php 페이지를 숨겨진 iframe에 띄울 것이다. 이때 검색할 아이디를 userid라는 쿼리를 포함하여 띄운다.

- idchk.php가 중복체크를 실행하여 중복된 아이디가 없다면 chkbool의 값이 1이 될 것이다. 0이라면 아직 중복체크를 실행하지 않았거나 중복된 아이디이므로 경고창을 띄우고 프로그램을 끝낸다. 만약 chkbool의 값이 0이 아니라면(1이라면) 중복체크를 실행하여 중복된 아이디가 없다는 뜻이므로 입력된 데이터들을 join_insert.php로 보낸다.

1.2 아이디 중복체크

아이디 중복체크는 join.php 내의 iframe 내에서 실행되는 페이지로 사용자 화면에서는 보이지 않도록 숨겨져 있다. join.php의 스크립트에 의하여 사용자가 입력한 아이디를 쿼리로 받아들여 이를 member 테이블에서 중복되는 아이디가 있는지 검사하여 그 여부를 알려준다.

idchk.php

```php
<?php
include "common.php";
$id = $_GET['userid'];
if($id == ""){
    echo "<script>
            alert('아이디를 입력하세요.');
            parent.document.join.chkbool.value = 0;
            parent.document.join.chkbtn.value = '중복체크';
          </script>";
    exit;
}
$data = mysqli_query($conn,"SELECT * FROM member WHERE userid='$id'");
$num = mysqli_num_rows($data);

if($num > 0){
    echo "<script>
            alert('사용할 수 없는 아이디입니다.');
            parent.document.join.chkbool.value = 0;
            parent.document.join.chkbtn.value = '중복체크';
          </script>";
}else{
    echo "<script>
            alert('사용 가능한 아이디입니다.');
            parent.document.join.chkbool.value = 1;
            parent.document.join.chkbtn.value = '사용가능';
          </script>";
}
?>
```

설명

- GET방식 쿼리로 받아온 userid를 변수로 저장한다.
- 아이디를 입력하지 않았을 때 경고창을 띄우고 아이프레임의 부모, 즉 join.php의 bool이라는 input에 0 값을 넣는다. 버튼의 텍스트는 '중복체크'라고 쓰고 아랫줄이 실행되지 않고 끝나도록 exit한다.
- member 테이블에서 해당 아이디를 검색하여 검색된 결과의 행(레코드) 수를 알아낸다.
- 검색된 결과가 있다면(중복된 아이디) 그 결과의 레코드 수는 0보다 클 것이다. 만약 그렇다면 '사용할 수 없는 아이디'라고 경고창을 띄우고 현재 아이프레임의 부모, 즉 join.php의 bool이라는 input에 0 값을 넣고 버튼의 텍스트는 '중복체크'라고 쓴다.
- 결과의 레코드 수가 0보다 크지 않다면 검색된 결과가 없다는 뜻이므로 '사용가능한 아이디'라고 경고창을 띄우고 bool의 값을 1값을 넣어주고 버튼의 텍스트를 '사용가능'이라고 바꿔준다.

1.3 회원가입 데이터 저장

join.php로부터 받아온 데이터를 member 테이블에 저장한다.

join_insert.php

```php
<?php
include "common.php";
$userid = $_POST['userid'];
$userpw = $_POST['userpw'];
$username = $_POST['username'];
$userpw = password_hash($userpw, PASSWORD_DEFAULT, ['cost' => 12]);




$data = mysqli_query($conn,"INSERT INTO member
(userid, userpw, username, userlv) VALUES
('$userid','$userpw','$username',1)");


echo "<script>
        alert('회원가입이 완료되었습니다.');
        location.href='login.php';
      </script>";
?>
```

설명

- POST방식으로 받아온 userid, userpw, username을 변수로 저장한다.
- 입력한 비밀번호는 관리자도 알 수 없도록 암호화 하여 저장해야 한다. 이때 password_hash라는 함수를 이용하여 텍스트를 해쉬화(암호화)하는데 첫 번째 인수를 원본텍스트, 두 번째 인수를 암호화 알고리즘, 세 번째 인수를 암호화 가중치로 지정한다. 가중치 숫자가 높을수록 높은 수준의 암호화가 가능하나 암호화/복호화 속도가 현저히 느려진다.
- 암호화된 비밀번호를 포함한 나머지 데이터들을 member 테이블에 저장한다. 이때 신규 회원의 레벨은 1로 지정한다. 이로써 레벨이 1인 회원과 레벨이 0인 비회원을 구분할 수 있게 된다.
- 회원가입을 완료하고 로그인 페이지로 이동한다.

2. login.php (로그인/로그아웃 페이지)

2.1 로그인

사용자의 아이디와 비밀번호를 입력받아 login_insert.php로 넘겨서 아이디와 비밀번호의 유효성을 검사한다.

login.php	설명
<?php include "common.php"; include "header.php"; ?> <form name="login" action="login_insert.php" method="post"> <input type="text" name="userid" /> <input type="password" name="userpw" /> <button type="submit">로그인</button> 회원가입 </form> <?php include "footer.php";?>	• userid와 userpw를 입력받아 post 방식으로 login_insert.php로 보낸다.

2.2 로그인 확인

login.php로부터 받아온 userid를 이용해 member 테이블에서 해당하는 아이디가 존재하는지 검사를 하고 입력한 비밀번호인 userpw와 member 테이블에 있는 암호화된 비밀번호와 서로 비교하여 올바른 비밀번호를 입력했는지 확인할 수 있다. 올바른 아이디와 비밀번호를 입력한 경우 세션을 생성하여 로그인이 돼 있다는 사실을 기록한다.

login_insert.php	설명
<?php include "common.php"; $userid = $_POST['userid']; $userpw = $_POST['userpw']; $data = mysqli_query($conn,"SELECT * FROM member WHERE userid='$userid'"); $len = mysqli_num_rows($data); if($len == 0){ echo "<script>alert('ID가 존재하지 않습니다.');history.back();</script>"; }else{ $row = mysqli_fetch_array($data); $veri = password_verify($userpw, $row['userpw']); if($veri){ $_SESSION['userid'] = $userid; $_SESSION['userlv'] = $row['userlv']; $_SESSION['username'] = $row['username']; echo "<script>alert('로그인에 성공했습니다.');history.go(-2);</script>"; }else{ echo "<script>alert('비밀번호가 일치하지 않습니다.');history.back();</script>"; } } ?>	• post방식으로 userid와 userpw를 입력받는다. • member 테이블에서 해당 아이디가 존재하는지 확인하기 위해 데이터를 검색하고 행 개수를 파악한다. • 데이터의 행 개수가 0이라면 데이터가 존재하지 않는다는 뜻이므로 경고창을 띄운다. • 사용자가 입력한 비밀번호와 member 테이블 내에 있는 암호화된 비밀번호 값을 비교하는 함수인 password_verify함수를 사용한다. 사용자가 입력한 비밀번호를 hash함수로 암호화 한 후 암호화된 비밀번호끼리 비교하는 것이 아니라 암호화되지 않은 원본 텍스트와 암호화 된 텍스트를 비교하는 것이다. • 비교 후 두 텍스트가 동일한 것으로 확인되었다면 password_verify함수는 true를 반환한다. 이때 userid와 userlv, username을 세션으로 등록해 둔다. 추후 이 세션들은 로그인 여부, 게시판 별 권한 확인 등에서 쓰인다.

2.3 로그아웃

logout.php를 열게 될 경우 모든 세션들을 삭제하고 로그아웃을 진행하게 된다.

logout.php	설명
<?php include"common.php"; session_destroy(); echo"<script>alert('로그아웃 되었습니다.');history.back();</script>"; ?>	• 존재하는 모든 세션을 삭제한다. userid, userlv, username을 삭제하므로 로그인하지 않은 계정으로 취급된다.

3. 리스트형 게시판 만들기

1. write.php (게시물 글쓰기 페이지)

1.1 mysql과 연결하기

게시물의 제목을 나열하여 리스트의 형식으로 보여주는 리스트형 게시판의 글쓰기 페이지이다. 각 게시물의 번호와 제목, 내용, 작성자, 작성일, 조회수, 답글의 no, 비밀글 여부 등을 저장할 것이다. 본 예제에서는 DB명을 [vision], 게시판의 테이블명을 [board]라고 명명하였다. [board]테이블의 스키마는 아래와 같다.

[board]

속성명	no	title	content	writer	id	date	count	reply	sec	file	pass
종류	INT	VARCHAR	TEXT	VARCHAR	VARCHAR	DATE	INT	INT	BIT	VARCHAR	VARCHAR
길이	10	200	-	100	100	-	10	10	1	100	300
데이터 정렬방식	-	UTF8UNICODE	UTF8UNICODE	UTF8UNICODE	UTF8UNICODE	-	-	-	-	UTF8UNICODE	UTF8UNICODE
설명	PRIMARY값으로 각 게시물을 특정하는 유일값	게시물 제목	게시물 내용	작성자 이름	작성자 아이디	작성일	조회수	답글 테이블에 있는 답글의 no	비밀글 여부	첨부 파일명	비회원용 게시물 비밀번호

1.2 권한 설정

write.php는 쿼리명령 없이 단독으로 사용하지 않는다. write.php?bo_ta=notice 등과 같이 어느 테이블에 게시물을 작성할지 테이블명을 쿼리문으로 명시하여 사용한다.

config.php

```php
<?php
$notice_list = 0;
$notice_read = 0;
$notice_write = 1;

$adminlv = 9;
$bo_num = 10;
$bo_block = 10;
?>
```

write.php

```php
<?php
include "common.php";
include "header.php";
$bo_ta = $_GET['bo_ta'];

$bo_admin = false;
$bo_write = false;


if($_SESSION['userlv'] >= $adminlv){
    $bo_admin=true;
}
if($_SESSION['userlv'] >= ${$bo_ta."_write"}){
    $bo_write=true;
}


if(!($bo_write || $bo_admin)){
    echo "<script>alert('글쓰기 권한없음');
                history.back();</script>";
    exit;
}

if(isset($_SESSION['userlv']) != 0){
    $wr_namehide = $_SESSION['userid'];
    $wr_name = $_SESSION['username'];
    $readonly = "readonly";
}else{
    $wr_namehide = "";
    $wr_name = "";
    $readonly = "";
}
// 아래에서 내용 이어짐
```

설명

- write.php가 실행될 때 GET 쿼리문으로 받았던 bo_ta의 값을 저장한다. 이것이 게시물을 저장할 테이블 명이 된다.
- 본 게시판에서 관리자 권한, 쓰기 권한을 설정하기 위한 값이다. 초기값은 false를 넣어두고 사용자의 레벨이 지정된 권한을 만족한다면 true로 바꿔줄 것이다. 즉 $bo_admin이 true라면 관리자 권한을 획득한 것이다.
- 현재 로그인한 사용자의 레벨이 config.php에서 설정된 관리자 레벨 이상이라면 $bo_admin의 값을 참으로 바꾼다.

- 현재 로그인한 사용자의 레벨이 config.php에서 설정된 글쓰기 가능 레벨 이상이라면 $bo_write의 값을 참으로 바꾼다. 게시판 별로 권한이 다를 수 있으므로 게시판 테이블 이름에 따라 ${$bo_ta."_write"}와 같이 변수명을 제작한다. 예를 들어 현재 테이블 이름이 notice라면 $notice_write 라는 변수가 만들어 진다.
- 관리자가 아니고 글쓰기 권한이 없는 사용자라면 경고창을 띄우고 페이지를 벗어난다.

- 사용자의 레벨이 0이 아니라면(로그인 한 사용자라면) 표면적으로 보여줄 작성자를 사용자의 이름으로 지정하고 내부적으로 저장할 이름을 사용자의 아이디로 지정한다. 그리고 작성자 input 태그의 내용을 바꿀수 없게 readonly 속성을 추가할 것이다.
- 그렇지 않다면(로그인을 안한 사용자라면) 작성자를 본인이 직접 작성할 수 있도록 비워두고 readonly 속성을 추가하지 않을 것이다. 그러면 사용자가 직접 작성자 이름을 쓸 수 있다.

1.3 데이터 입력 폼

데이터를 입력 받을 폼을 제작한다. 내용을 입력받아 write_insert.php로 보내는데 이때 파일 첨부를 하기 위해서 post방식으로 전송해야만 한다. 암호화 형식은 "multipart/form-data"로 설정한다. 표면적으로 표시되는 작성자는 사용자의 이름이지만 작성자 본인확인에서는 사용자의 아이디로 판별을 해야 하므로 내부적으로 작성자의 아이디를 저장한다. 로그인을 하지 않은 사용자라면 별도로 게시물에 비밀번호를 입력하도록 폼을 생성시킨다. 비로그인 사용자가 작성한 글을 작성한 본인이 수정, 삭제할 때 이 비밀번호를 통해 인증을 하게 된다.

write.php	설명
`// 위에서 내용 이어짐` `<form id="writeform" action="write_insert.php"` `method="post" enctype="multipart/form-data">` 작성자 :	• 첨부 파일을 업로드하기 위해서 반드시 POST방식을 써야한다. 암호화 형식은 폼의 데이터와 첨부파일을 같이 보내므로 "multipart/form-data"를 사용한다.
`<input type="text" name="writer" <?php echo $readonly;?>` `value="<?php echo $wr_name;?>" />`	• 로그인 사용자라면 사용자의 이름을 writer의 value로 기입하고 readonly속성을 추가하여 읽기전용으로 만든다. 비 로그인 사용자라면 본인이 직접 기입하도록 한다.
`<input type="hidden" name="writerid"` `value="<?php echo $wr_namehide;?>" />`	• 로그인 사용자라면 사용자의 아이디를 writerid의 value로 기입하고 비 로그인 사용자라면 비워둔다.
`<input type="hidden" name="table"` `value="<?php echo $bo_ta;?>" />`	• 이 폼에서 write_insert.php로 데이터를 보낼 때 저장할 테이블 명을 미리 기입해 둔다. 사용자 아이디와 테이블명 폼은 사용자들에게 보여줄 필요가 없으므로 type은 hidden으로 지정한다.
`<?php` `if($_SESSION['userlv'] == 0 && $bo_write){` `echo "비밀번호 : ";` `echo "<input type='password' name='notuserpw' />";` `}` `?>`	• 본 게시판에서 비 로그인 사용자도 글을 쓸 수 있도록 설정되어 있을 때 비 로그인 사용자가 추후 글을 수정하거나 삭제할 때 비밀번호가 필요하므로 비밀번호를 입력받는 폼을 생성한다.
제목 : `<input type="text" name="title" />`	• 제목을 입력하는 폼을 생성한다.
비밀글 : `<input type="checkbox" name="secrete" value="on" />`	• 비밀글 여부를 지정하는 체크박스 폼을 생성한다.
본문 : `<textarea name="content"></textarea>`	• 본문을 입력하는 textarea 폼을 생성한다.
첨부파일 : `<input type="file" name="myfile" />`	• 첨부파일을 올리는 폼을 생성한다.
`<button onclick="history.back();" type="button">취소</button>`	• 취소버튼을 누르면 이전 페이지로 돌아간다.
`<button type="submit">글쓰기</button>` `</form>`	• 최종적으로 데이터를 write_insert.php로 전송한다.
`<?php` `include "footer.php";` `?>`	

2. write_insert.php (데이터베이스 입력 페이지)

2.1 전송받은 데이터 변수화 하기

write.php가 POST방식으로 보내온 데이터를 변수로 저장한다. 작성일은 현재 날짜로 자동으로 만들어지므로 따로 받아올 필요가 없다. 로그인 사용자라면 그 사용자의 이름으로 작성자를 설정하고 아이디는 내부적으로 따로 저장해 둔다. 비 로그인 사용자라면 본인이 직접 입력한 데이터가 작성자의 값이 된다. 만약 비밀글을 체크했다면 그 값을 통해 비밀글 여부를 지정한다. 비 로그인 사용자가 입력한 비밀번호는 그대로 저장하지 않고 password_hash 함수를 통해 암호화하여 저장한다. 업로드 한 파일이 있다면 그 파일의 확장자가 업로드가 가능한 확장자인지 용량이 지나치게 크지 않은지 확인하고 파일명이 겹치지 않도록 새 이름을 부여한 후 지정된 폴더에 저장한다. 실제 파일은 이와 같이 저장하고 데이터베이스에는 해당 파일의 이름만 저장해 둔다. 사용자가 첨부 파일을 업로드하면 그 파일은 서버 내 임의의 임시 저장소에 보관된다. 여기에 보관된 파일은 일정 시간이 지난 후 자동으로 삭제되므로 서버에 저장하기 위해서는 적당한 경로와 이름을 지정하여 옮겨야 한다.

write.php	설명
<?php include "common.php"; $bo_ta = $_POST['table']; $title = $_POST['title']; $content = $_POST['content']; $date = date("Y-m-d"); if($_POST['writerid'] == ""){$writer = $_POST['writer'];} else{$writer = $_POST['writerid'];}	• 데이터를 입력할 테이블 명을 변수에 저장한다. • 게시물의 제목을 변수에 저장한다. • 게시물의 내용을 변수에 저장한다. • 현재 날짜를 2019-05-21의 형식으로 저장한다. • 작성자의 id를 저장하기로 한 폼이 비어있다면 비로그인 사용자라는 뜻이므로 이때에는 작성자의 id를 사용자가 직접 입력한 이름으로 지정한다. 작성자의 id를 저장하기로 한 폼이 비어있지 않다면 로그인 사용자라는 뜻이므로 이때에는 사용자의 아이디를 그대로 작성자의 id로 지정한다.
$secrete = $_POST['secrete']; if($secrete == "on"){$secrete = 1;} else{$secrete = 0;}	• 비밀글에 체크를 했다면 그 값이 "on"이 되므로 이를 통해 그 여부를 알 수 있다. secrete 값이 1이면 작성자 본인과 관리자만 볼 수 있는 글로 설정될 것이다.
$notuserpw = $_POST['notuserpw']; if($notuserpw != ""){ $notuserpw = password_hash($notuserpw, PASSWORD_DEFAULT, ['cost'=>10]); }	• 비 로그인 사용자라면 별도로 비밀번호를 입력했을 것이다. 이 값을 그대로 저장하지 않고 password_hash 함수를 통해 암호화 한 후 저장할 것이다.
$filename = $_FILES['myfile']['name']; $filetype = $_FILES['myfile']['type']; $filetemp = $_FILES['myfile']['tmp_name']; $filesize = $_FILES['myfile']['size'];	• 업로드한 파일의 이름 • 업로드한 파일의 파일형식(확장자) • 업로드한 파일이 임시 저장된 위치 • 업로드한 파일의 사이즈
if($filename!=""){ if($filetype=="text/html"||$filetype=="text/htm"\|\|$filetype=="text/php"|| ~ ||$filetype=="text/jsp"\|\|$filetype=="text/phtml"||$filetype=="text/js"){ echo "<script> alert('업로드 할 수 없는 파일 형식입니다.'); history.back(); </script>"; exit; }	• 파일을 업로드 했다면 허용할 수 없는 확장자인지 확인한다. 본 사이트를 공격할 수 있는 여지가 있는 웹 문서 확장자들을 사전에 차단한다. 각종 악성코드 및 인젝션 공격 등을 막기 위해 각종 언어가 들어있는 이러한 웹문서를 차단해야 한다.
if($filesize > 3145728){ echo "<script> alert('3MB 이상의 파일은 업로드할 수 없습니다.'); history.back(); </script>"; exit; } // 아래에서 내용 이어짐	• 파일 사이즈를 체크하여 지정된 용량 이상 업로드 할 수 없도록 차단한다. 단위는 byte단위이며 3,145,728byte는 3MB이다. 서버의 저장 공간과 트래픽 용량은 무한하지 않으므로 업로드 파일의 용량을 제한해야 한다.

2.2 첨부파일 업로드하기 & 데이터베이스 저장하기

write.php

```
// 위에서 내용 이어짐

    $name = explode(".", $filename);
    $nameLen = count($name);
    $file_ext = $name[$nameLen-1];

    $file_name = "file_".time().".".$file_ext;
    $file_name = addslashes($file_name);
    $file_name = strtolower($file_name);



    $full_url = "/.../html/upload/".$file_name;


    move_uploaded_file($filetemp, $full_url);
}else{
    $filename = null;
}
$data = mysqli_query($conn,"INSERT INTO $bo_ta
(title, content, writer, date, count, file,
reply, secrete, pass)
VALUES
('$title', '$content', '$writer', '$date', 0, '$file_name',
'', $secrete, '$notuserpw')");



echo "<script>alert('게시물이 등록되었습니다.');
            history.go(-2);</script>"
?>
```

```
$name = "jiho.photo.jpg"
↓
$name = ["jiho","photo","jpg"]
          [0]    [1]    [2]
$name[2] ⇒ "jpg"
```

설명

- 파일명을 "."(dot)으로 나누어 파일명과 확장자로 나눈다. explode함수는 그 결과를 배열변수로 반환한다. 이 배열에서 마지막 값이 확장자이므로 이 값을 추출한다. (배열의 개수 – 1는 마지막 번호임)
- 타임스탬프를 이용해 서로 겹치지 않을 파일명을 생성하고 그 뒤에 위에서 만든 확장자를 붙인다. 따옴표와 같은 특수문자가 들어있었을 가능성이 있으므로 addslashes 함수를 써서 "를 \"의 형태로 바꾼다. strtolower함수를 통해 모든 대문자를 소문자로 바꾼다.
- 서버 내 따로 마련해둔 업로드 폴더 경로를 지정하고 만들어진 파일명을 붙여서 전체 경로를 만든다. 서버 경로는 호스팅업체에 문의한다.
- move_uploaded_files 함수를 통해 임시폴더에 있는 파일을 지정된 위치에 지정된 이름으로 이동한다.
- 파일을 업로드 하지 않았다면 데이터베이스에 저장할 파일명은 비워둔다.
- 지정된 테이블에 위 데이터들을 저장하는 SQL 명령어이다. 사용자가 입력한 데이터 이외에 날짜는 현재 날짜가 값이 되고 조회수는 0으로 저장한다. 답글도 아직 없으므로 공백으로 저장한다.
- SQL 명령어 사용 시 문자열 데이터는 반드시 따옴표로 묶고 숫자 데이터는 그렇지 않다는 사실에 유의하여야 한다.
- 저장이 완료된 후 [뒤로 가기]를 두 번 실행한다. 두 번을 하는 이유는 한 번만 하게 되면 write.php로 돌아가기 때문이다.

3. board.php (게시물 리스트 페이지)

3.1 테이블, 페이지 지정하기 & 권한 확인하기

board.php는 쿼리명령 없이 단독으로 사용하지 않는다. board.php?bo_ta=notice&pagi=1 등과 같이 어느 테이블에 몇 번째 페이지를 볼 지 테이블명과 페이지 번호를 쿼리문으로 명시하여 사용한다. (pagi = pagination)

PHP

```
<?php
include "common.php";
include "header.php";
$bo_ta = $_GET['bo_ta'];
$pagi = $_GET['pagi'];
if(!isset($pagi) || $pagi == ""){
   $pagi = 1;
}
if(!isset($bo_ta)){
   echo "<script>alert('잘못된 접근입니다.');
                history.back();</script>";
   exit;
}

// 아래에서 내용 이어짐
```

설명

- board.php가 실행될 때 쿼리로 테이블명을 받아서 그 값을 지정한다.
- 마찬가지로 pagi 번호를 받아서 그 값을 알아낸다.
- 만약 그 값이 없거나 비어있다면 1이라고 지정해 둔다.
- 테이블명이 지정되지 않으면 접근을 막는다.

PHP	설명
// 위에서 내용 이어짐 if($_SESSION['userlv'] < $notice_read){ echo "<script>alert('게시판을 열람할 권한이 없습니다.'); location.href='index.php';</script>"; exit; } if($_SESSION['userlv'] >= $adminlv){ $bo_admin=true; } if($_SESSION['userlv'] >= $notice_write){ $bo_write=true; }	• 사용자의 레벨이 게시판에 설정된 읽기 권한보다 작으면 경고창을 띄우고 페이지를 이탈한다. • 사용자의 레벨이 관리자 권한 레벨 이상이면 본 게시판에서 관리자 권한을 부여한다. • 사용자의 레벨이 글쓰기 레벨 이상이면 본 게시판에서 글쓰기 권한을 부여한다.

4. 갤리리형 게시판 만들기