図 1　Wiki の形とトップページ

図 2　一覧表示

図 3　一覧表示（管理設定画面）

図 4　新規投稿（１）


タイトル☆
メニュー
記事
トップページ
一覧表示
新規投稿
新着記事2件
ここがタイトル
サイドメニュー（右）
新着コメント1件
ここがコメントの本...
検索
検索
設定
設定
編集
タイトル：新規投稿テスト
本文：
プレビュー
送信
サイドメニュー（右）
サイドメニュー右側。
[設定]
↓
[その他設定]
↓
[サイドメニュー表示データ]
から右側のサイドメニューに表
示する記事を決定させる。
初期設定ではid2のこの記事を
選択。
記事とは別に、各ページで利用
されるメソッド一覧を表示させ
ることも可能。
同じくその他設定の画面から設
定のこと。
Copyright 2014 Mr.Nanashi © All Rights Reserved.

図 5　新規投稿（２）

図 6　コメントの返信

図 7　表示項目の設定

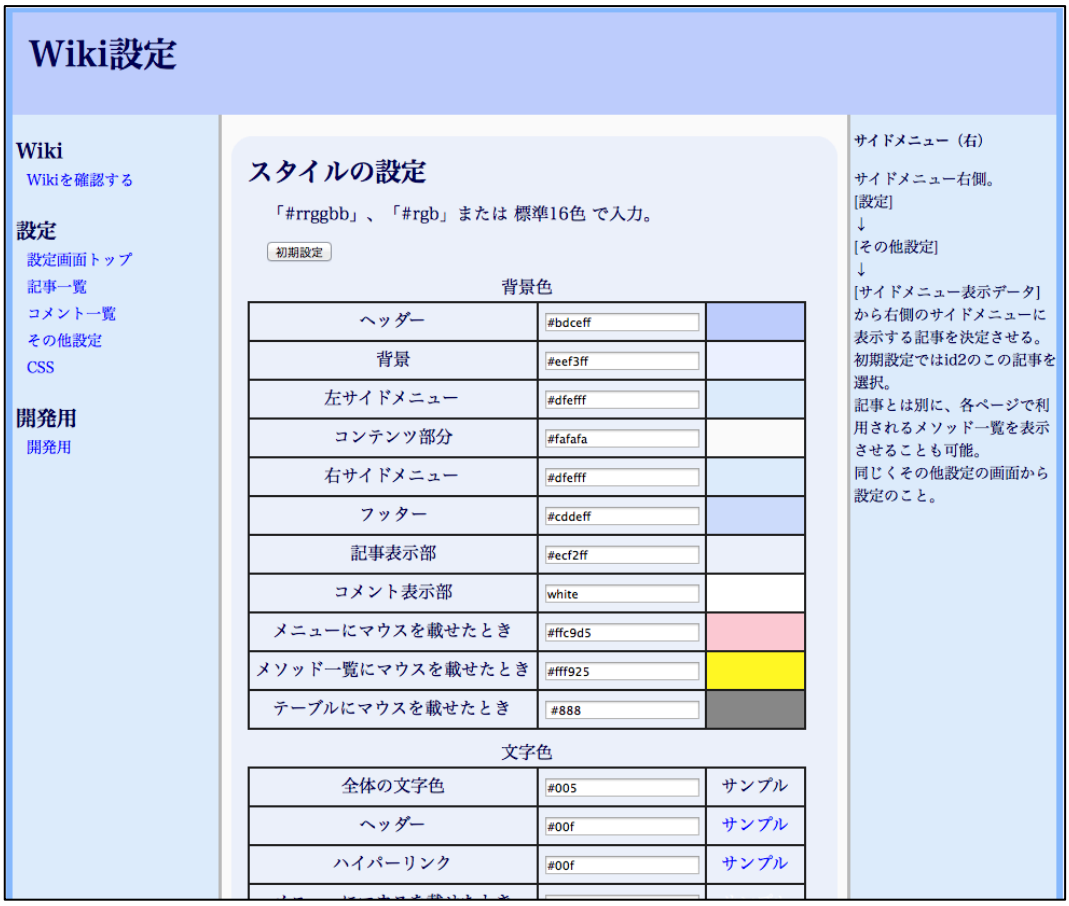

図 8　スタイル CSS の設定

図 9　スタイルの設定（エラー発生時）

定数名	フィールド名	テーブル名
ID	id	kiji
TITLE	title	
HONBUN	honbun	
HIDUKE	hiduke	
COM_ID	com_id	comment
KIJI_ID	kiji_id	

表 1　Cols クラスの定数とデータベースの関連

記事全体 $kiji	記事 1 $kiji[0]	ID	$kiji[0][cols::ID]
		タイトル	$kiji[0][cols::TITLE]
		本文	$kiji[0][cols::HONBUN]
		日付	$kiji[0][cols::HIDUKE]
	記事 2 $kiji[2]	ID	$kiji[1][cols::ID]
		タイトル	$kiji[1][cols::TITLE]
		本文	$kiji[1][cols::HONBUN]
		日付	$kiji[1][cols::HIDUKE]
	記事 3 $kiji[2]	ID	$kiji[2][cols::ID]
		タイトル	$kiji[2][cols::TITLE]
		本文	$kiji[2][cols::HONBUN]
		日付	$kiji[2][cols::HIDUKE]

表 2　　記事データにおける配列内の配列の例

【database クラス】

```
const DatabaseName = "wiki_oop";       //仕様するデータベースの定義
public function setDatabase($database)// データベース名セッティング
{
        $this->database = $database;
}
        public function connect()              //データベース接続
{
        $database_name = database::DatabaseName;
        $connect = mysql_connect("localhost","root","");
        if (! $connect) {
                die("データベースに接続できませんでした。");
        }
        mysql_select_db("$database_name");
}
        public function close()                //データベース接続停止
{
        mysql_close();
}
        public function query($sql)            // クエリの実行
{
        $data = mysql_query($sql);
        return $data;
}
```

【cols クラス】

```
const ID = 0;       //kiji.id
const TITLE = 1;    //kiji.title
const HONBUN = 2;   //kiji.honbun
const HIDUKE = 3;   //kiji.hiduke
// ↑記事 ↓コメント
const COM_ID = 4;   // comment.id
const KIJI_ID = 5; //comment.kiji_id
const NAME = 6;     //comment.name
const COM_HONBUN = 7;         // comment.com_honbun
const COM_HIDUKE = 8;         //comment.hiduke
const REPLY_ID = 9;//comment.hiduke
//        ↑コメント↓セッティング
const SETTINGKEY = 10;        //setting.SettingKey
const SETTINGVALUE = 11;              //setting.SettingValue
```

【base クラス】

```
public $setting;

public function p($data)      //単純に p で括られた文　id、class は利用しない
{
        return "<p>$data</p>¥n";
}
public function p_class($data,$class) //p に class を設定する場合
{
        return "<p class='$class'>$data</p>¥n";
}
public function formStart($action)
{
        return "<form action='$action' method='POST'>¥n";
}
public function formEnd(){
        return "</form>¥n";
}
```

```
public function input($type,$name,$value)                    //通常の input
{
          return "<input type='$type' name='$name' value='$value' />";
}
public function input_text($name,$value,$size)  //inpu type=text でサイズが指定可
{
          return "<input type='text' name='$name' value='$value' size='$size' />";
}
public function input_del($value,$function,$data)
{
          return "<input type='button' name='del' value='$value'
onclick='$function(this,¥"$data¥")' />";
}
public function textarea($name,$col,$row,$textarea)
{
          return "<textarea name='$name' cols='$col' rows='$row'>$textarea</textarea>";
}
public function h1($h1)
{
          return "<h1>$h1</h1>¥n";
}
public function h2($h2)
{
          return "<h2>$h2</h2>¥n";
}
public function h3($h3)
{
          return "<h3>$h3</h3>¥n";
}
public function h4($h4)
{
          return "<h4>$h4</h4>¥n";
}
public function a($href,$text)
{
          return "<a href='$href'>$text</a>";
}
public function hr()
{
          return "<hr>¥n";
}
public function ulStart()
{
          return "<ul>¥n";
}
public function ulEnd()
{
          return "</ul>¥n";
}
public function li($li)
{
          return "<li>$li</li>¥n";
}
public function tableStart()
{
          return "<table>¥n";
}
public function tableEnd()
```

```
{
        return "</table>¥n";
}
public function trStart()
{
        return "<tr>¥n";
}
public function trEnd()
{
        return "</tr>¥n";
}
public function th($th)
{
        return "¥t<th>$th</th>¥n";
}
public function td($td)      //通常の td
{
        return "¥t<td>$td</td>¥n";
}
public function tdClass($class , $td) //td に class を設定する場合
{
        return "¥t<td class='$class'>$td</td>¥n";
}
public function divStart($id)          //div に id を設定する場合
{
        return "<div id='$id'>¥n";
}
public function divClassStart($id) //div に class を設定する場合
{
        return "<div class='$id'>¥n";
}
public function divEnd(){
        return "</div>¥n";
}
```

【system クラス】

```
public $id;
public $header;
public $menu;
public $function_list;                  //関数一覧
public $file_name;
public $css_data;
const StyleSheet = "system/StyleSheet.css";              //css ファイルの定義
const StyleSheetC = "system/copy_StyleSheet.css";        //css ファイルの定義
const JavaScript = "system/JavaScript.js";               // javascript ファイルの定義
const FILE_NAME = "system/system.php";          // このファイル
const FILE_REQUIRE = "/^require_once/";                  // require_once
const FILENAME_RESEARCH = '/[a-zA-Z]*¥.php/';   // ファイル名の取得
const METHOD_LIST = "/^[¥t]*public function/";  // publicfunction で始まるもを取得
const GET_METHOD = "/-(>|&gt;){1}[a-zA-Z0-9_]*/";        //メソッド名の取得
const SET_SPAN = "/¥/¥/.*$/m";                  //「//」を探す
const SET_SPAN2 = "(/¥*/?(¥n|[^/]|[^*]/)*¥*/)"; //「//」を探す
const CSS_SELECTER = "/^[a-zA-Z#,.¥*: 0-9]*/";           //css セレクターの取得
const CSS_SELECTER_COLS = "/{/";                         //css セレクター列の選択
const CSS_BGCOLOR = "/background-color[ ]?:[#a-z0-9 ]*/"; //background-color を探す
```

```
const CSS_COLOR = "/[¥t]color[ ]?:[#a-z0-9 ]*/";          //color を探す

public function __construct()          // system クラス生成時に自動的に実行。ID の受信
{
          if(isset($_GET['id'])) {
                    $id = $_GET['id'];
                    $this->id = $id;
          } else if (isset($_POST['id'])) {
                    $id = $_POST['id'];
                    $this->id = $id;
          } else {
                    $id = mysql_insert_id();
                    $this->id = $id;
          }
          $this->file_name = basename($_SERVER['PHP_SELF']);// 現在のファイル名を取得
          $this->setting = $this->getSettingData();
          $this->getFileName();
}

public function SearchMethod($data)              // メソッドを検索取得する。
{
          if ( preg_match( system::GET_METHOD, $data )) {
                    preg_match_all( system::GET_METHOD, $data, $method_name);
          }
          $i = 0;
          foreach ( $method_name[0] as $value ) {
                    $method_array[$i] = str_replace("->" , "" , $value);
                    $i++;
          }
          return $method_array;
}

public function getMethodContents($method)       // 与えられたメソッド名の中身を返す
{
          $function_list = $this->function_list;
          foreach ( $function_list as $key => $value )
          {
                    if ( $key == $method ) {
                              $return_data = $value;
                              break;
                    }
          }
          return $return_data;
}

public function MethodContentsSearch($method)   //別のメソッドがあるか探す
{
          $data = $this->getMethodContents($method);
          $method_name = array();
$var_name =
array( "id","header","menu","function_list","file_name","css_data","setting","method_shosa
i");
          if ( preg_match( system::GET_METHOD, $data ) ) {
                    preg_match_all( system::GET_METHOD, $data, $method_name);
                    $i = 0;
                    foreach ( $method_name[0] as $value ) {
                              $value = str_replace("&amp;nbsp;" , "asdfasdf;" , $value);
                              $value = str_replace("->" , "" , $value);
```

```
                                $value = str_replace("-&gt;" , "" , $value);
                                if(! in_array($value, $var_name)) {
                                        $method_array[$i] = $value;
                                }
                                $i++;
                        }
                } else {
                        $method_array = FALSE;
                }
                return $method_array;
}

public function methodJudge()           // メソッドを判定して処理を決める
{
                $now = $this->getFileData(); // 今開いている php ファイルのソースを取得
$c = 0;
                $method_array = $this->SearchMethod($now);
                $data = array();
                $check = array();
                for ( $i=0;$i<count($method_array);$i++) {
                        $kensaku[] = $method_array[$i];
                        for ( $j = 0; $j < count($kensaku); $j++ ) {
                                if ( !in_array($kensaku[$j] , $check) ) {
                                        $check[] = $kensaku[$j];
                                        $data = $this->MethodContentsSearch($kensaku[$j]);
                                        $c++;
                                        if ( is_array($data)) {
                                                foreach ( $data as $key => $val2) {
                                                        if ( $val2 != "" ) {
                                                                $kensaku[] = $val2;
                                                        } else {
                                                unset($data);
                                                        }
                                                }
                                        }
                                }
                        }
                }
                $kensaku = array_unique($kensaku);
                return $kensaku;
}


public function getAllMethod()
{
                $data = $this->getFileData();
                $method_array = $this->SearchMethod($data);

                $function_list = $this->function_list;
                foreach ( $function_list as $list_key => $list_value )
                {
                        foreach ($method_array as $use_key => $use_value )
                        {
                                if ( $list_key == $use_value )
                                {
                                        $new_data = $this->SearchMethod($list_value);
                                }
                        }
```

```
        }
}

public function dispMethod()
{
        $data = $this->getFileData();
        $method_array = $this->SearchMethod($data);
        $this->getFileName();
        $function_list = $this->function_list;
        foreach ( $function_list as $list_key => $list_value )
        {
                foreach ($method_array as $use_key => $use_value )
                {
                        if ( $list_key == $use_value )
                        {
                                print "<hr>" . $list_value;
                        }
                }
        }
}

public function getFileData() //      現在接続しているページのソースコードを取得。
{
        $file_name = $this->file_name;
        $data = "";
        $file = fopen($file_name, "r");
        if ( $file == false ) {
                print "ERR.ファイルのオープンに失敗";
        } else {
                while ( $line = fgets($file)) {
                        $data .= nl2br($line);
                }
        }
        return $data;
}

public function getFileName()                    // 使用されるファイル名取得関数生成
{
        $system_file = system::FILE_NAME;
        $file = fopen($system_file, "r");
        if ( $file == false ) {
                print "ERR.ファイルのオープンに失敗";
        } else {
                $filename_array[] = $system_file;
                while ( $line = fgets($file)) {
                        if ( preg_match( system::FILE_REQUIRE, $line ) ) {
                                preg_match( system::FILENAME_RESEARCH, $line,
$file_name);
                                foreach ( $file_name as $value ) {
                                        $filename_array[] = "system/" . $value;
                                }
                        }
                }
        }
        $this->setFunctionList($filename_array);
}

public function setFunctionList($filename_array)          // 関数の一覧を作成する
```

```
{
        $function_array = array();
        foreach( $filename_array as $value )
        {
                $data = $this->setFunctionData($value);
                foreach ( $data as $array_data_key => $array_data_value )
                {
                        if ( strstr($array_data_value, "¥t")) {
                        $array_data_value = str_replace("¥t" ,
" ",$array_data_value);
                        }
                        $function_array[$array_data_key] =
nl2br(htmlentities($array_data_value, ENT_QUOTES, "UTF-8"));
                }
        }
        $this->function_list = $function_array;
}

public function setFunctionData($file_name)             // 一覧から配列を作る
{
        $function_array = array();
        $file = fopen($file_name, "r");
        if ( $file == false ) {
                print "ファイルのオープンに失敗しました";
        } else {
                while ( $line = fgets($file) ) {
                        if ( preg_match( system::METHOD_LIST , $line ) ) {
                                list ( $function , $comment ) = explode("¥t//", $line);
                                $function = str_replace("public function ", "",
$function);
                                list ( $function_name, $argument) =
explode("(",$function);
                                $function_name = str_replace("¥t" , "" ,
$function_name);
                                $function_array[$function_name] = $line;
                        } else {
                                $function_array[$function_name] .= $line;
                        }
                }
        }
        return $function_array;
}

public function getSettingData()       // データベース利用でのセッティング
{
        $sql="select * from setting;";
        $data = mysql_query($sql);
        while($record = mysql_fetch_array($data)){
                $SettingData[$record[SettingKey]] = $record[SettingValue];
        }
        return $SettingData;
}

public function getTitleList()         // 記事データのタイトルのみを取得する
{
        $sql = "select title,id from kiji;";
        $data = mysql_query($sql);
        while ( $record = mysql_fetch_array($data)) {
```

```
                $title[$record[id]] = $record[title];
        }
        return $title;
}

public function input_require($data)
{
        if ( strlen($data) == 0 ) {
                print $this->p("ERR：入力されていません。");
        }
}

public function index()      //トップページ用変形
{
        $TopPageID = $this->setting[topID];
        $contents = $this->getDataId($TopPageID);
print <<< eot
{$this->divClassStart("index")}
{$this->shosai($contents)}
{$this->divEnd()}
eot;
}

public function arrange($data)        //整形用。未使用。
{
        return $data;
}

public function SetHeader($header)    //ヘッダーのセット
{
        $this->header = $header;
}

public function SetMenu($menu)                  //メニューのセット
{
        $this->menu = $menu;
}

public function getWikiData()         //すべての記事を日付昇順に取得
{
        $data = $this->getData("kiji.hiduke");
        return $data;
}

public function getData($field)       //記事データ全件取得
{
        $sql="select * from kiji order by $field;";
        $kiji = $this->getKiji($sql);
        return $kiji;
}
public function getDataId($id)        //ID で記事データを取得
{
        $sql="select * from kiji where id=$id;";
        $kiji = $this->getKiji($sql);
        return $kiji[0];
}

public function getKiji($sql)         // sql でデータベースから記事データを取り出す
```

```
{
        $data = mysql_query($sql);
        while ( $record = mysql_fetch_array($data)) {
                $kiji[cols::ID] = $record["id"];
                $kiji[cols::TITLE] = $record["title"];
                $kiji[cols::HONBUN] = $record["honbun"];
                $kiji[cols::HIDUKE] = $record["hiduke"];
                $kiji_data[] = $kiji;
        }
        return $kiji_data;
}

public function getCommentKijiId($id) //記事 ID ごとにコメントを取得
 {
        $sql = "select * from comment where kiji_id = $id;";
        $data = mysql_query($sql);
        return $data;
}

public function getCommentId($id)     //コメント ID ごとにコメントを取得
{
        $sql = "select * from comment where id = $id;";
        $data = mysql_query($sql);
        if( $record = mysql_fetch_array ( $data )) {
                $Comment[cols::KIJI_ID] = $record['kiji_id'];
                $Comment[cols::NAME] = $record['name'];
                $Comment[cols::COM_HONBUN] = $record['com_honbun'];
                $Comment[cols::COM_HIDUKE] = date('Y/m/d', $record["hiduke"]);
                return $Comment;
        }
}

public function GetCommentList($limit)//コメントの一覧を取得
{
        $sql = "select kiji.title , comment.id , comment.kiji_id , comment.com_honbun ,
comment.hiduke , comment.name  from kiji , comment where kiji.id = comment.kiji_id order by
comment.hiduke desc $limit;";
        $data = mysql_query($sql);
        while($record = mysql_fetch_array($data)){
                $CommentData[cols::TITLE] = $record["title"];
                $CommentData[cols::COM_ID] = $record["id"];
                $CommentData[cols::KIJI_ID] = $record["kiji_id"];
                $CommentData[cols::COM_HONBUN] = $record["com_honbun"];
                $CommentData[cols::COM_HIDUKE] = $record["hiduke"];
                $CommentData[cols::NAME] = $record["name"];
                $CommentList[] = $CommentData;
        }
        return $CommentList;
}

public function getAllCommentNumber()          // コメントが全部で何件あるかを取得
{
        $sql = "select id from comment;";
        $data = mysql_query($sql);
        $number = mysql_num_rows($data);
        return $number;
}
```

```
public function getAllKijiNumber()              // 記事が全部で何件あるかを取得
{
        $sql = "select id from kiji;";
        $data = mysql_query($sql);
        $number = mysql_num_rows($data);
        return $number;
}

public function getCommentNumber($id) //コメントの件数を取得
{
        $data = $this->getCommentKijiId($id);
        $com_number = mysql_num_rows($data);
        return $com_number;
}

public function createSearchSql($sw, $subject)  //検索で用いる sql の繰り返し処理
{
        $i = 1;
        $kensu = count($sw);
        foreach($sw as $val ) {
                $sql .= "$subject like '%$val%'";
                if( $i < $kensu ) {
                        $sql .= " and ";
                }
                $i++;
        }
        return $sql;
}

public function SearchData() // and 検索で用いるスペースの処理
{
        $sw=$_POST['sw'];
        $sw = str_replace("　", " ", $sw);
        $sw = explode( " " , $sw );
        $sql_kiji = "select * from kiji where ";
        //ここから where 検索条件
        $sql_kiji .= $this->createSearchSql($sw , "title");
        $sql_kiji .= " or ";
        $sql_kiji .= $this->createSearchSql($sw , "honbun");
        //ここまで where 検索条件
        $sql_kiji .= " order by hiduke desc;";
        $sql_com .= "select * from comment,kiji where ";
        //ここから where 検索条件
        $sql_com .= $this->createSearchSql($sw , "comment.com_honbun");
        //ここまで where 検索条件
        $sql_com .=" and kiji.id = comment.kiji_id order by comment.hiduke desc;";
        $data_kiji = mysql_query($sql_kiji);
        $record_num_kiji = mysql_num_rows($data_kiji);
        $data_com = mysql_query($sql_com);
        $record_num_com = mysql_num_rows($data_com);
        print $this->h2("検索結果");
        print "<p>「";
        foreach ( $sw as $val ) {
                print " " . $val . " ";
        }
        print "」で検索した結果、" . $record_num_kiji . "件の記事と、" . $record_num_com .
"件のコメントが見つかりました。</p>";
        if ($record_num_kiji != 0 or $record_num_com != 0) {
```

```
                    while($record_kiji = mysql_fetch_array($data_kiji)) {
                              $SearchData[cols::ID] = $record_kiji['id'];
                              $SearchData[cols::TITLE] = $record_kiji['title'];
                              $SearchData[cols::HONBUN] = $record_kiji['honbun'];
                              $SearchData[cols::HIDUKE] = $record_kiji['hiduke'];
                              $KijiData[] = $SearchData;
                    }
                    while ( $record_com = mysql_fetch_array($data_com)) {
                              $SearchComData[cols::COM_ID] = $record_com['id'];
                              $SearchComData[cols::TITLE] = $record_com['title'];
                              $SearchComData[cols::KIJI_ID] = $record_com['kiji_id'];
                              $SearchComData[cols::NAME] = $record_com['name'];
                              $SearchComData[cols::COM_HONBUN] = $record_com['com_honbun'];
                              $SearchComData[cols::COM_HIDUKE] = $record_com['hiduke'];
                              $CommentData[] = $SearchComData;
                    }
                    $ReturnData['kiji'] = $KijiData;
                    $ReturnData['comment'] = $CommentData;
                    return $ReturnData;
          }
}

public function CommentArrange($data) //コメント表示用の整形。先頭 22 字で切り取り
{
          $data = mb_strimwidth($data, 0, 22, "..." , utf8);
          return $data;
}

public function SetupArrange($data) // セットアップ表示用の整形。先頭 30 字で切り取り
{
          $ComData = mb_strimwidth($data, 0, 30, "……" , utf8);
          return $ComData;
}

public function index_arrange($data)              // トップページ用の整形。未使用か
{
          $indexhonbun = $this->arrange($data);
          $indexhonbun = mb_strimwidth($indexhonbun, 0, 320, " ・・・<br><br>……(省略＃＃
処理を付け加えよ)……<br><br>", utf8);
          return $indexhonbun;
}

public function shosai($kiji)                     // 詳細表示用の整形
{
          $id = $kiji[cols::ID];
          $ContentsData .= $this->h3($this->a("shosai.php?id=$id" , $kiji[cols::TITLE]));
          $ContentsData .= $this->p(nl2br($this->arrange($kiji[cols::HONBUN])));
          $ContentsData.=$this->p_class("最終更新日:". date('Y/m/d', $kiji[cols::HIDUKE]),
"time");
          return $ContentsData;
}

public function Menu_NewData()                    // メニューの新着記事を取得
{
          $NewArticle = $this->setting["NewArticle"];
          $new_kiji = $this->getData("kiji.hiduke desc limit $NewArticle");
          $kiji_num = $this->getAllKijiNumber();
          if ( $NewArticle > $kiji_num ) {
```

```
                    $NewArticle = $kiji_num;
          }
          $data .= $this->h4("新着記事" . $NewArticle . "件");
          $data .= $this->ulStart();
          foreach($new_kiji as $row) {
                    $data .= $this->li($this->a("shosai.php?id={$row[cols::ID]}",
"{$row[cols::TITLE]}"));
          }
          $data .= $this->ulEnd();
          return $data;
}

public function Menu_NewComment()                // メニューの新着コメントを取得
{
          $NewArticle = $this->setting["NewArticle"];
          $new_comment = $this->GetCommentList("limit $NewArticle");
          $com_num = $this->getAllCommentNumber();
          if ( $NewArticle > $com_num ) {
                    $NewArticle = $com_num;
          }
          $data .= $this->h4("新着コメント" . $NewArticle . "件");
          $data .= $this->ulStart();
          foreach($new_comment as $row) {
                    $data .= $this->li($this->a("shosai.php?id={$row[cols::KIJI_ID]}",
"{$this->CommentArrange($row[cols::COM_HONBUN])}"));
          }
          unset($new_comment);
          $data .= $this->ulEnd();
          return $data;
}

public function Menu_NewPost()                   // メニューの記事関係リスト表示
{
return <<<eot
{$this->h4("記事")}
{$this->ulStart()}
{$this->li("{$this->a("index.php","トップページ")}")}
{$this->li("{$this->a("list.php","一覧表示")}")}
{$this->li("{$this->a("edit.php","新規投稿")}")}
{$this->ulEnd()}
eot;
}

public function Menu_Setup() // メニューのセットアップリスト表示
{
return <<<eot
{$this->h4("設定")}
{$this->ulStart()}
{$this->li("{$this->a("Setup.php","設定")}")}
{$this->ulEnd()}
eot;
}


public function prev_jushin()          // 記事編集画面→プレビュー画面でのデータ受信
{
          $prev_data[cols::TITLE] = $_POST['title'];
          $prev_data[cols::HONBUN] = $_POST['honbun'];
```

```
		$prev_data[cols::ID] = $_POST['id'];
		$prev_data[cols::HIDUKE]=time();
		date_default_timezone_set("Asia/Tokyo");
		return $prev_data;
}

public function find_css($data, $kind)		//cssの中からbackground-colorのものを探し、色指定部分のみを返す
{
		if(preg_match( $kind, $data)) {
			preg_match ( $kind , $data, $bgcolor);
		} else {
			$bgcolor = 0;
		}
		list ( $string , $color ) = explode ( ":" , $bgcolor[0]);
		return rtrim($color);
}

public function set_css($data,$color,$kind)		//送られてきたデータ（背景色）をbackground-colorにセットする
{
		if ( $kind == system::CSS_BGCOLOR ) {
			$color = "background-color:" . $color;
		} else if ( $kind == system::CSS_COLOR ) {
			$color = "¥tcolor:" . $color;
		}
		$return = preg_replace($kind , $color , $data );
return $return;
}

public function CommentJudge($id)	// コメントの件数から表示方法を判定
{
		$judge = $this->getCommentNumber($id);
		$disp = $this->h3("コメント");
		$disp .= $this->hr();
		if ( $judge != 0 ) {
			$disp .=$this->CommentDisp($id);
		} else {
			$disp .=$this->NoComment();
		}
		return $disp;
}

public function setCssData($file_name)
{
		$file = fopen($file_name, "r");
		if ( $file == false ) {
			print $this->p("ファイルのオープンに失敗しました。");
		} else {
			$css_array = array();
			while ( $line = fgets($file)) {
				if ( $line != "" ) {
					if ( preg_match( system::CSS_SELECTER_COLS, $line )) {
						preg_match( system::CSS_SELECTER, $line,
$selecter);
						$selecter_name = $selecter[0];
						$css_array[$selecter_name] .= $line;
					} else {
```

```
                                                $css_array[$selecter_name] .= $line;
                                        }
                                }
                        }
                $this->css_data = $css_array;
                }
}

public function setbgcolor_key()        //background-color を変更する要素の一覧を作成。
{
        $bg = array ( "div#header"=>"ヘッダー",
                "body"=>"背景",
                "div#menu"=>"左サイドメニュー",
                "div#contents"=>"コンテンツ部分",
                "div#source"=>"右サイドメニュー",
                "div#footer"=>"フッター",
                "div.index"=>"記事表示部",
                "div#comment"=>"コメント表示部",
                "div#menu li a:hover"=>"メニューにマウスを載せたとき",
                "div#source li a:hover"=>"メソッド一覧にマウスを載せたとき",
                "table td a:hover"=>"テーブルにマウスを載せたとき"
        );
        return $bg;
}

public function setcolor_key()          //color(文字色)を変更する要素の一覧を作成。
{
        $color = array (
                "body"=>"全体の文字色",
                "div#header a"=>"ヘッダー",
                "a"=>"ハイパーリンク",
                "div#menu li a:hover"=>"メニューにマウスを載せたとき",
                "div#source li a:hover"=>"メソッド一覧にマウスを載せたとき",
                "table td a:hover"=>"テーブルにマウスを載せたとき"
        );
        return $color;
}
```

【disp クラス】

```
public function NormalHeader()          // ヘッダーの整形
{
        $title = $this->setting[WikiTitle];
        return $this->h1("{$this->a('index.php', "{$title}")}");
}
public function footer()                // フッターの整形
{
        $footer = $this->setting[footer];
        return $this->p("{$footer}");
}

public function SetupHeader()           // セットアップ画面のヘッダー
{
        $title = $this->setting[WikiTitle];
        return $this->h1("Wiki 設定");
}
```

```
public function setSideMenu_r() {
          $id = $this->setting[SideMenuRight];
          if ( $id == 0 ) {
                    $mes .= "<h3>使用されるメソッド一覧</h3>";
                    $method_array = $this->methodJudge();
                    $mes .= "<ul>";
                    $page = $this->file_name;
                    foreach ( $method_array as $key => $val ) {
                              $mes .= "<li><a
href='method.php?kind=$val&from=$page'>$val</a></li>¥n";
                    }
                    unset($method_array);
                    $mes .= "</ul>";
          } else {
                    $kiji = $this->getDataId($id);
                    $title = $kiji[cols::TITLE];
                    $honbun = nl2br($kiji[cols::HONBUN]);
                    $mes .= "<h3>$title</h3>";
                    $mes .= "<p>$honbun</p>";

          }
          print $mes;
}

public function HtmlStart($title)                // HTML 開始部から body 開始まで
{
$stylesheet = System::StyleSheet;
$javascript = System::JavaScript;
date_default_timezone_set('Asia/Tokyo');
if ( $_GET['from'] ) {
          $from = $_GET['from'];
          $this->file_name = $from;
}
print <<< eot
<!DOCTYPE html>
<html lang="ja">
<head>
<title>$title</title>
<meta charset='utf-8' />
<link rel='stylesheet' href="$stylesheet" />
<script src="$javascript"></script>
</head>
<body>
eot;
}

public function HtmlEnd()    // HTML 終了部。フッターの表示も含む
{
print <<< eot
</div>
<div id="source">
eot;
$this->setSideMenu_r();
print <<< eot
</div>    <!-- end of #source-->
</div>    <!-- end of #main-->
<div id="footer">
```

```
{$this->footer()}
</div>    <!-- end of #footer-->
</div>    <!-- end of wrapper-->
</body>
</html>
eot;
}         // ここまで htmlEnd

public function editArrange($data)              //編集表示時のデータの整形
{
          return $data;
}

public function new_page()             // 新規記事投稿時の画面
{
          $title_list = $this->getTitleList();
          $title_list = implode("," , $title_list);
          $data = "";
          $data .= $this->divClassStart("index");
          $data .= $this->h2("新規投稿");
          $data .= $this->p("新しく投稿する記事のタイトルを入力し、送信ボタンを押してくださ
い。");
          $data .= $this->formStart("edit.php");
          $data .= $this->p("タイトル：{$this->input("text" , "title" , "")}");
          $data .= "<input type='hidden' id='title_list' name='title_list'
value='$title_list'>";
          $data .= "<p><input type='button' value='送信' onclick='return
title_check(this.form)' /></p>";
          $data .= $this->formEnd();
          $data .= $this->divEnd();
          print $data;
}

public function edit($url)   // 編集メニュー
{
          $id = $this->id;
          $kiji_data = $this->getDataId($id);
          $kiji = $this->editArrange($kiji_data);
          $title_list = $this->getTitleList();
          unset($title_list[$kiji[cols::ID]]);
          $title_list = implode("," , $title_list);
print <<< eot
{$this->h2("編集")}
{$this->divClassStart("index")}
{$this->formStart("{$url}?id=$id")}
{$this->p("タイトル：{$this->input("text" , "title" , "{$kiji[cols::TITLE]}")}")}
{$this->p("本文：{$this->textarea("honbun" , "75" ,"35", "{$kiji[cols::HONBUN]}")}")}
{$this->input("hidden" , "id" , "$id")}
eot;
print "<input type='hidden' id='title_list' name='title_list' value='$title_list'>";
print "<p><input type='submit' name='prev' value='プレビュー'>";
print "<input type='button' value='送信' onclick='return title_check(this.form)' /></p>";
print <<< eot
{$this->formEnd()}
{$this->divEnd()}
eot;
}
```

```
public function SetupEdit($url)        // 管理画面での記事編集表示部
{
          $id = $this->id;
$this->edit($url);
print <<< eot
{$this->formStart("SetupList.php")}
{$this->input("hidden" , "topID" , "$id")}
{$this->p("{$this->input("submit" ,"TopPage" , "この記事をトップページに設定")}")}
{$this->formEnd()}
{$this->formStart("SetupList.php")}
{$this->input("hidden", "id" , "$id")}
{$this->p_class("{$this->input_del("記事削除","del_check_article", "$id")}" , "")}
{$this->formEnd()}
eot;
}

public function method_shosai()                    // メソッド詳細の表示
{
          $kind = $_GET['kind'];
          $function_list = $this->function_list;
          $disp = "";
          foreach ( $function_list as $key => $value ) {
                    if ( $key == $kind ){
                              if ( strstr ( $value, "&amp;nbsp;")) {
                                        $value = str_replace ( "&amp;nbsp;" ,
"    " , $value );                                }
                              if ( strstr($value,"//")){
                                        $value = preg_replace(system::SET_SPAN , "<span
class='red'>$0</span>", $value);
                              }
                              if ( strstr($value, "/*")) {
                                        $value = preg_replace(system::SET_SPAN2 , "<span
class='red'>$0</span>", $value);
                              }
                              $disp = $value;
                              break;
                    }
          }
          unset($key);
          print $this->divClassStart("index");
          print $this->h2("{$kind}メソッド");
          print "<p>$value</p>";
          print $this->divEnd();
}

public function shosai_disp()          // 記事の詳細表示
{
          $id = $this->id;
          $kiji = $this->getDataId($id);
print <<< eot
{$this->divClassStart("index")}
{$this->shosai($kiji)}
{$this->divStart("comment")}
{$this->CommentJudge($id)}
{$this->CommentPost($id)}
{$this->divEnd()}{$this->divEnd()}
eot;
}
```

```
public function sort()                  // 一覧表示での並べ替え用&lt;select&gt;
{
          $sort = "kiji.hiduke";
          if( isset($_POST['sort'])) {
                    $sort = $_POST['sort'];
          }
          print $this->formStart("");
          print "<select name='sort'>";
          $list = array("kiji.hiduke" => "日付昇順" , "kiji.hiduke desc" => "日付降順" ,
"kiji.title" => "タイトル昇順" , "kiji.title desc" => "タイトル降順" , "kiji.id" => "ID 昇順
" , "kiji.id desc" => "ID 降順");
          foreach ( $list as $key => $val ) {
                    if ( $sort != $key ) {
                              print "<option value=¥"$key¥">$val</option>¥n";
                    } else {
                              print "<option value=¥"$key¥" selected>$val</option>¥n";
                    }
          }
          print "<input type='submit' value='送信' />";
          print "</select>";
          print $this->formEnd();
          return $sort;
}

public function ichiran()    // 記事の一覧表示
{
print <<< eot
{$this->h2("記事一覧")}
eot;
          $sort = $this->sort();
          $WikiData = $this->getData($sort);
print <<< eot
{$this->tableStart()}
{$this->trStart()}
{$this->th("ID")}
{$this->th("タイトル")}
{$this->th("最終更新日")}
{$this->th("コメント")}
{$this->th("編集")}
{$this->trEnd()}
eot;
          foreach( $WikiData as $row){
                    $com_number = $this->getCommentNumber($row[cols::ID]);
                    $hiduke = date('Y 年 m 月 d 日' , $row[cols::HIDUKE]);
                    $id = $row[cols::ID];
print <<< eot
{$this->trStart()}
{$this->td("{$row[cols::ID]}")}
{$this->td("{$this->a("shosai.php?id={$row[cols::ID]}" , $row[cols::TITLE])}")}
{$this->td("{$hiduke}")}
{$this->td("{$this->a("shosai.php?id={$row[cols::ID]}", $com_number . " 件")}")}
{$this->td("{$this->formStart('edit.php')}{$this->input('hidden','id',$id)}{$this->input('
submit','edit','編集')}{$this->formEnd()}")}
{$this->trEnd()}
eot;
          }         //foreach 終
print <<< eot
```

```
{$this->tableEnd()}
<!--{$this->divEnd()}-->
eot;
}

public function SetupIchiran()          // 管理画面での一覧表示
{
          print $this->h2("記事一覧");
          $sort = $this->sort();
          $WikiData = $this->getData("$sort");
print <<< eot
{$this->tableStart()}
{$this->trStart()}
{$this->th("ID")}
{$this->th("TOP")}
{$this->th("記事タイトル")}
{$this->th("最終更新日")}
{$this->trEnd()}
eot;
          foreach($WikiData as $row ) {
                    $hiduke = date('Y年m月d日', $row[cols::HIDUKE]);
print <<< eot
{$this->trStart()}
{$this->tdClass("id", "{$row[cols::ID]}")}
eot;
                    print "<td class='top'>";
                    if($this->setting[topID] == $row[cols::ID] ) {
                              print "☆";
                    }
                    print "</td>";
print <<< eot
{$this->td("{$this->a("SetupShosai.php?id={$row[cols::ID]}", $row[cols::TITLE])}")}
{$this->td("{$hiduke}")}
{$this->trEnd()}
eot;
          }
print <<< eot
{$this->tableEnd()}
{$this->p("※[ ☆ ]がついた記事は、トップページに登録されています。")}
<!--{$this->divEnd()}-->
eot;
}

public function setupTop()              // 管理画面トップ
{
print <<< eot
{$this->h2("設定")}
{$this->divStart("index")}
{$this->tableStart()}
{$this->trStart()}
{$this->th("設定画面トップ")}
{$this->td("このページ")}
{$this->trEnd()}
{$this->trStart()}
{$this->th("記事一覧")}
{$this->td("トップページの設定")}
{$this->trEnd()}
{$this->trStart()}
```

```
{$this->th("コメント一覧")}
{$this->td("コメントの一覧表示（削除）")}
{$this->trEnd()}
{$this->trStart()}
{$this->th("その他設定")}
{$this->td("ヘッダー・フッター・新着記事表示件数の設定")}
{$this->trEnd()}
{$this->trStart()}
{$this->th("CSS 設定")}
{$this->td("スタイルシートの編集")}
{$this->trEnd()}
{$this->tableEnd()}
{$this->divEnd()}
eot;
}

public function setupList()  // タイトル・フッター・新着記事表示件数の設定
{
          $setting = $this->setting;
$kiji = $this->getData("id");
print <<< eot
{$this->h3("設定")}
{$this->divStart("index")}
{$this->formStart("SetupWiki.php")}
{$this->p("Wiki タイトル：タイトルとして使用される文字列。")}
{$this->p_class("{$this->input_text("WikiTitle", "$setting[WikiTitle]", "80")}",
"setupother")}
<hr>
{$this->p("フッター：Wiki 最下部のフッターに表示させる文字列。")}
{$this->p_class("{$this->input_text("footer", "$setting[footer]", "80")}", "setupother")}
<hr>
{$this->p("新着記事 表示件数：サイドメニューの新着記事の表示件数を設定。")}
{$this->p_class("<input type='number' name='NewArticle' value='$setting[NewArticle]' />",
"setupother")}
<hr>
{$this->p("右サイド表示データ：右側のサイドメニューに表示するデータの選択。")}
<p class="setupother">
<select name='SideMenuRight'>
<option value="0">未選択</option>
eot;
foreach  ( $kiji as $row ) {
$id = $row[cols::ID];
$title = $row[cols::TITLE];
if ( $setting[SideMenuRight] == $id ) {
          print "<option value='$id' selected>$title</option>¥n";
} else {
          print "<option value='$id'>$title</option>¥n";
}
}
print <<< eot
</select>
</p>
<hr>
<p>
<input type="button" value="更新" onclick="return setting_check(this.form)" />
<input type="reset" value="初期値" />
</p>
{$this->formEnd()}
```

```
{$this->divEnd()}
eot;
}

public function frame()      // ボックス構造の生成
{
print <<< eot
<div id="wrapper">
<div id="header">
{$this->{"$this->header"}()}
</div>
<div id="main">
<div id="menu">
{$this->{"$this->menu"}()}
</div>
<div id="contents">
eot;
}

public function Menu_Search()          // サイドメニューの検索窓の作成
{
return <<< eot
{$this->h4("検索")}
{$this->formStart("search.php")}
{$this->p($this->input("text" ,"sw" , ""))}
{$this->p($this->input("submit" ,"search" , "検索"))}
{$this->formEnd()}
eot;
}

public function search()     // 検索の実施
{
          $disp = "";
          $data = $this->SearchData();
          $KijiData = $data['kiji'];
          $CommentData = $data['comment'];
          if ( $KijiData != "" ) {
                    $disp .= $this->h3("記事");
                    $disp .= $this->tableStart();
                    $disp .= $this->trStart();
                    $disp .= $this->th("ID");
                    $disp .= $this->th("タイトル");
                    $disp .= $this->th("本文");
                    $disp .= $this->th("日付");
                    $disp .= $this->trEnd();
                    foreach ( $KijiData as $row ){
                              $hiduke = date('Y年m月d日', $row[cols::HIDUKE]);
                              $title = $row[cols::TITLE];
                              $id = $row[cols::ID];
                              $honbun = $this->SetupArrange($row[cols::HONBUN]);
                              $disp .= $this->trStart();
                              $disp .= $this->td("{$row[cols::ID]}");
                              $disp .= $this->td("{$this->a("Shosai.php?id={$id}" ,
$title)}");
                              $disp .= $this->td("{$honbun}");
                              $disp .= $this->td("{$hiduke}");
                              $disp .= $this->trEnd();
                    }
```

```
                    $disp .= $this->tableEnd();
                    print $disp;
                    $disp ="";
          }
          if ( $CommentData != "" ) {
                    print $this->h3("コメント");
                    $disp .= $this->tableStart();
                    $disp .= $this->trStart();
                              $disp .= $this->th("ID");
                              $disp .= $this->th("投稿者名");
                              $disp .= $this->th("コメント本文");
                              $disp .= $this->th("投稿日");
                              $disp .= $this->th("記事タイトル");
                    $disp .= $this->trEnd();
                    foreach ( $CommentData as $row ){
                              $hiduke = date('Y年m月d日', $row[cols::COM_HIDUKE]);
                              $com_id = $row[cols::COM_ID];
                              $name = $row[cols::NAME];
                              $kiji_id = $row[cols::KIJI_ID];
                              $com_honbun = $this->SetupArrange($row[cols::COM_HONBUN]);
                              $disp .= $this->trStart();
                              $disp .= $this->td("{$com_id}");
                              $disp .= $this->td("{$name}");
                              $disp .=
$this->td("{$this->a("Shosai.php?id={$kiji_id}#comment", $com_honbun)}");
                              $disp .= $this->td("{$hiduke}");
                              $disp .= $this->td("{$this->a("Shosai.php?id={$kiji_id}",
$title)}");
                              $disp .= $this->trEnd();
                    }
                    $disp .= $this->tableEnd();
                    print $disp;
          }
}

public function NormalMenu()          //通常サイドメニュー
{
          $NewArticle = $this->setting["NewArticle"];
          $data = $this->h3("メニュー");
          $data .= $this->Menu_NewPost();
          if ( $NewArticle > 0 ) {
                    $data .= $this->Menu_NewData();
                    $data .= $this->Menu_NewComment();
          }
          $data .= $this->Menu_Search();
          $data .= $this->Menu_setup();
          return $data;
}

public function SetupMenu()           //設定画面サイドメニュー
{
return <<< eot
{$this->h3("Wiki")}
{$this->ulStart()}
{$this->li("{$this->a("index.php","Wiki を確認する")}")}
{$this->ulEnd()}
{$this->h3("設定")}
{$this->ulStart()}
```

```
{$this->li("{$this->a("Setup.php","設定画面トップ")}")}
{$this->li("{$this->a("SetupList.php","記事一覧")}")}
{$this->li("{$this->a("SetupComList.php","コメント一覧")}")}
{$this->li("{$this->a("SetupWiki.php","その他設定")}")}
{$this->li("{$this->a("SetupCss.php","CSS")}")}
{$this->ulEnd()}
{$this->h3("開発用")}
{$this->ulStart()}
{$this->li("{$this->a("system/develop.php","開発用")}")}
{$this->ulEnd()}
eot;
}

public function dispSetupContents()    // 記事登録完了画面
{
          $registData = $this->prev_jushin();
print <<< eot
{$this->p("更新しました！")}
{$this->divClassStart("index")}
{$this->shosai($registData)}
{$this->divEnd()}
{$this->p($this->a("SetupList.php" , "戻る"))}
eot;
}

public function prev()                 // 記事プレビュー画面
{
          $prevData = $this->prev_jushin();
print <<< eot
{$this->h2("プレビュー")}
{$this->p("現時点では、この記事はまだ記事は投稿されていません！")}
{$this->divClassStart("index")}
{$this->shosai($prevData)}
{$this->divEnd()}
{$this->p("<input type='button' onclick='history.back();' value='戻る' />")}
eot;
}

public function CommentList()          // コメントの一覧表示（管理画面）
{
          $data = $this->GetCommentList("");
          $disp = $this->h2("コメント一覧");
          $disp .= $this->tableStart();
          $disp .= $this->trStart();
                    $disp .= $this->th("投稿者名");
                    $disp .= $this->th("コメント本文");
                    $disp .= $this->th("投稿日");
                    $disp .= $this->th("記事タイトル");
          $disp .= $this->trEnd();
          foreach ( $data as $row ) {
                    $name = $row[cols::NAME];
                    $com_id = $row[cols::COM_ID];
                    $com_honbun = $this->SetupArrange($row[cols::COM_HONBUN]);
                    $hiduke = date('Y/m/d', $row[cols::COM_HIDUKE]);
                    $kiji_id = $row[cols::KIJI_ID];
                    $title = $row[cols::TITLE];
                    $disp .= $this->trStart();
                    $disp .= $this->td("{$name}");
```

```
                    $disp .= $this->td("{$this->a("SetupComShosai.php?id={$com_id}",
$com_honbun)}");
                    $disp .= $this->td("{$hiduke}");
                    $disp .= $this->td("{$this->a("SetupShosai.php?id={$kiji_id}" ,
$title)}");
                    $disp .= $this->trEnd();
          }
$disp .= $this->tableEnd();
print $disp;
}

public function CommentDisp($id)       // 詳細画面でのコメント部分
{
          $data = $this->getCommentKijiId($id);
          while ($record = mysql_fetch_array($data)) {
                    $id = $record['id'];
                    $reply_id = $record['reply_id'];
                    $name = $record['name'];
                    $com_honbun = $record['com_honbun'];
                    if ( strstr($com_honbun, ">>") && $reply_id != 0 ) {
                              $com_honbun = preg_replace("/>>.*さん/m" , "<span id='id$id'
onMouseover='openPopUp($reply_id)' onMouseout='closePopUp($reply_id)'>$0</span><span
id='mes$id' class='hidden'>てすと</span>", $com_honbun);
                    }
                    $CommentData .= "<div id='$id'>";
                    $CommentData .= "<h4 id='$id'>$name</h4>";
                    $CommentData .= $this->p(nl2br($com_honbun));
                    $CommentData .= "<p><input type=¥"button¥" value=¥"返信¥"
onclick=¥"setReply('$name','$id')¥"></p>";
                    $CommentData .= $this->p_class("投稿日：" . date('Y/m/d',
$record['hiduke']) , "time");
                    $CommentData .= "</div>";
          }
          return $CommentData;
}

public function CommentShosai()        // 管理画面でのコメントの詳細表示
{
          $id = $this->id;
          $data = $this->getCommentId($id);
print <<< eot
{$this->h2("コメント詳細")}
{$this->divClassStart("index")}
{$this->h4($data[cols::NAME])}
{$this->p($data[cols::COM_HONBUN])}
{$this->p_class("投稿日：" . $data[cols::COM_HIDUKE] , "time")}
{$this->formStart("SetupComList.php")}
{$this->input("hidden" , "id" , "$id")}
<input type="button" name="del" value="削除" onclick="del_check(this)" />
{$this->formEnd()}
{$this->divEnd()}
eot;
}

public function NoComment()  //        まだ記事にコメントがない場合の表示
{
          return $this->p("このトピックにはまだコメントがありません。");
}
```

```
public function CommentPost($id)       // コメント投稿用の入力欄の生成
{
return <<< eot

{$this->h4("コメントの投稿")}
<form action="shosai.php?id=$id" name="comment_post" method="POST">
{$this->p("名前：{$this->input("text" , "CommentName" , "")}")}
<p>本文：<textarea name="CommentHonbun" cols="70" rows="7"></textarea></p>
<input type="hidden" name="reply_id" value="">
<input type="button" value="投稿" onclick="return com_post_check(this.form)" />
{$this->formEnd()}
eot;
}

public function EditCss()              // StyleSheet 編集画面
{
          $css_array = $this->css_data;
          $disp .= $this->formStart("SetupCss.php");
          $bg = $this->setbgcolor_key();
          $color = $this->setcolor_key();
          $css_bgcolor = system::CSS_BGCOLOR;
          $css_color = system::CSS_COLOR;

          foreach ( $bg as $key => $val) {
                    $bgcolor[$key] = $this->find_css($css_array[$key], $css_bgcolor);
          }
          foreach ( $color as $key => $val) {
                    $fontcolor[$key] = $this->find_css($css_array[$key], $css_color);
          }
          if ( $_POST ){
                    $data .= $this->p("<input type='button' onclick='history.back();' value='
やっぱやめ' />");
          }
          $data .= $this->divClassStart("index");
          $data .= $this->h2("スタイルの設定");
          $data .= $this->p("「#rrggbb」、「#rgb」または 標準 16 色 で入力。");
          $data .= "<form action='SetupCss.php' method='POST'>";
          $data .= "<input type='hidden' name='reset' />";
          $data .= "<p><input type='button' value='初期設定' onclick='reset_css(this)'
/></p>";
          $data .= "</form>";
          $data .= "<form name='setupcss' action='' method='POST'>";
          $data .= $this->tableStart();
          $data .= "<caption>背景色</caption>";
          foreach ( $bgcolor as $selecter => $value ) {
                    $data .= "<tr><td>$bg[$selecter]</td><td><input type='text'
name='bgcolor[$selecter]' value='$value'></td><td class='mihon'></td></tr>";
          }
          $data .= $this->tableEnd();
          $data .= $this->tableStart();
          $data .= "<caption>文字色</caption>";
          foreach ( $fontcolor as $selecter => $value ) {
                    $data .= "<tr><td>$color[$selecter]</td><td><input type='text'
name='color[$selecter]' value='$value'></td><td class='mihon'>サンプル</td></tr>";
          }
          $data .= $this->tableEnd();
          $data .= "<p><input type='button' value='送信' onclick='return css_check(this.form)'
```

```
/>";
          $data .= "<input type='reset' value='クリア' /></p>";
          $data .= "</form>";
          $data .= $this->divEnd();
          echo $data;
          echo "<script>getcssdata()</script>";
          echo "<p id='test'></p>";
}
```

【edit クラス】

```
public function write()                    //記事データの受信($_POST)
{
          $WikiData[cols::TITLE] = $_POST['title'];
          $WikiData[cols::HONBUN] = $_POST['honbun'];
          $WikiData[cols::HIDUKE]=time();
          return $WikiData;
}

public function CommentWrite()                         //コメンとデータの受信($_POST)
{
          $CommentData[cols::KIJI_ID] = $_GET['id'];
          $CommentData[cols::NAME] = $_POST['CommentName'];
          if ( strlen($CommentData[cols::NAME]) == 0 ) {
                    $CommentData[cols::NAME] = "名無し";
          }
          if ( $_POST['reply_id'] !="" ) {
                    $reply_id = $_POST['reply_id'];
                    $CommentData[cols::REPLY_ID] = $reply_id;
          } else {
                    $CommentData[cols::REPLY_ID] = 0;
          }
          $CommentData[cols::COM_HONBUN] = $_POST['CommentHonbun'];
          $CommentData[cols::COM_HIDUKE] = time();
          return $CommentData;
}

public function editArrange($kijiData)          // 整形
{
          return $kijiData;
}

public function regist()                   // 記事データのデータベース登録
 {
          $kijiData = $this->write();
          $WikiData = $this->editArrange($kijiData);
          $this->input_require($WikiData[cols::TITLE]);
          $sql="insert into kiji ( hiduke , title ) values ('". $WikiData[cols::HIDUKE] .
"' , '" . $WikiData[cols::TITLE] . "');";
          $touroku = new database;
          $data = $touroku->query($sql);
          if(! $data){
                    die("ERR : データが登録できませんでした。");
          } else {
//echo $this->p("登録完了！");
          }
```

```
}

public function CommentRegist()        //コメントのデータベース登録
 {
          $CommentData = $this->CommentWrite();
                    $sql="insert into comment (kiji_id, hiduke, name, com_honbun,
reply_id) values ('" . $CommentData[cols::KIJI_ID] . "' , '" .
$CommentData[cols::COM_HIDUKE] . "' , '" . $CommentData[cols::NAME] . "' , '" .
$CommentData[cols::COM_HONBUN] . "',  '" . $CommentData[cols::REPLY_ID] . "');";

          $touroku = new database;
          $data = $touroku->query($sql);
          if(! $data){
                    die("ERR：コメントできませんでした。");
          }
}


public function RemoveData()           //setting テーブルの内容書き換え
{
          $SettingData = $this->setting;
          $i = 0;
          $kensu = count($_POST);
          $sql = "insert into setting (SettingKey, SettingValue) values ";
          foreach ( $_POST as $key => $val ) {
                    if ( $i < $kensu ) {
                              $sql .= "('" . $key . "', '" . $val . "' )";
                              if ( $i < $kensu -1 ) {
                                        $sql .= ", ";
                              }
                    }
                    $i++;
          }
          $sql .= " on duplicate key update SettingValue=values(SettingValue);";
          $new_setting = new database;
          $data = $new_setting->query($sql);
          if(! $data){
                    die("データが更新できませんでした。");
          } else {
//                            echo "更新できました！";
          }
}

public function file_write($file_name,$data)     // データ登録
{
          $file = fopen($file_name, "w");
          if ( $file == false ) {
                    print $this->p("ファイルのオープンに失敗しました。");
          } else {
                    $written = fwrite($file, $data);
                    if ( $written == false ) {
                              print $this->p("ファイルの書き込みに失敗しました");
                    } else {
//                                        print $this->p("更新しました");
                    }
          fclose($file);
          }
}
```

```
public function UpdateCss()            // css のデータ更新
{
          $file_name = system::StyleSheet;
          $this->setCssData($file_name);
          extract($_POST);
          $css_data = $this->css_data;
          foreach ( $bgcolor as $key => $value ) {
                    $css_data[$key] = $this->set_css($css_data[$key] , $value ,
system::CSS_BGCOLOR);
          }
          foreach ( $color as $key => $value ) {
                    $css_data[$key] = $this->set_css($css_data[$key] , $value ,
system::CSS_COLOR);
          }
          $this->setCssData = $css_data;
          foreach ($css_data as $value ) {
                    $data .= $value;
          }
          $file_name = system::StyleSheet;
          $this->file_write($file_name, $data);
}

public function update()               // 記事データの更新
{
          $kiji_data = $this->write();
          $wiki_data = $this->treeArrange($kiji_data);
          $wiki_data[cols::ID] = $_POST['id'];
          $sql = "update kiji set hiduke='" . $wiki_data[cols::HIDUKE] . "' , title='" .
$wiki_data[cols::TITLE] . "' , honbun='" . $wiki_data[cols::HONBUN] . "' where id = '" .
$wiki_data[cols::ID] . "' ; ";
          $touroku = new database;
          $data = $touroku->query($sql);
          $data = mysql_query($sql);
          if(! $data){
                    die("データが更新できませんでした。");
          }
}

public function del()                  //記事の削除（トップ登録のものは不可）
{
          $id = $this->id;
          if ( $this->setting["topID"] == $id ) {
                    die("この記事はトップページに設定されているので削除できません！
");
          } else {
                    $sql = "delete from kiji where id=$id;";
                    $delete = new database;
                    $data = $delete->query($sql);
                    if(! $data) {
                              print "<p>削除できませんでした</p>";
                    } else {
                              print "<h2>削除完了</h2>";
                              print "<p>記事 id : {$id} は削除されました。</p>";
                              $this->CommentDelByKiji($id);
                    }
```

```
        }
}

public function CommentDelByKiji($id) //削除した記事に対応するコメントを削除
{
        $sql = "delete from comment where kiji_id = $id;";
        $delete = new database;
        $data = $delete->query($sql);
        if(! $data) {
                print $this->p("削除できませんでした");
        } else {
                echo $this->p("ついでにコメントも削除されました");
        }
}

public function CommentDel() // コメントの削除
{
        $id = $this->id;
        $sql = "delete from comment where id=$id;";
        $delete = new database;
        $data = $delete->query($sql);
        if(! $data) {
                print "<p>削除できませんでした</p>";
        } else {
                print "<h3>削除完了</h3>";
                print "<p>コメント id : {$id} は削除されました。</p>";
        }
}
```

【wiki.sql】 データベース登録用

```
#drop database wiki_oop;
create database wiki_oop;
use wiki_oop;
create table kiji (
          id integer unique primary key not null auto_increment,
          hiduke integer,
          title varchar(100),
          honbun text
);
insert into kiji ( hiduke, title, honbun) values ( unix_timestamp(), "ここがタイトル", "ここが本文。");
insert into kiji ( hiduke, title, honbun) values ( unix_timestamp(), "サイドメニュー（右）", "サイドメニュー右側。
[設定]
↓
[その他設定]
↓
[サイドメニュー表示データ]
から右側のサイドメニューに表示する記事を決定させる。
初期設定では id2 のこの記事を選択。
記事とは別に、各ページで利用されるメソッド一覧を表示させることも可能。
同じくその他設定の画面から設定のこと。");

create table comment (
          id integer unique primary key not null auto_increment,
          kiji_id text,
          hiduke integer,
          name text,
          com_honbun text,
          reply_id integer
);
insert into comment ( kiji_id, hiduke, name, com_honbun, reply_id) values ("1", unix_timestamp(), "投稿者名", "ここがコメントの本文です",0);

create table setting (
          SettingKey varchar(100) primary key not null,
          SettingValue text
);
insert into setting ( SettingKey, SettingValue) values ( "WikiTitle", "タイトル☆");
insert into setting ( SettingKey, SettingValue) values ( "footer", "Copyright 2014 Mr.Nanashi © All Rights Reserved.");
insert into setting ( SettingKey, SettingValue) values ( "topID", "1");
insert into setting ( SettingKey, SettingValue) values ( "NewArticle" , "5");
insert into setting ( SettingKey, SettingValue) values ( "SideMenuRight" , "2");
```

【index.php】 トップページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp();
$html->HtmlStart(INDEX);
$html->SetHeader(NormalHeader);
$html->SetMenu(NormalMenu);
```

```
$html->frame();
$disp = new disp();
$disp->index();
$html->HtmlEnd();
$database->close();
?>
```

【edit.php】 新規投稿・編集ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp;
$html->HtmlStart(編集);
$html->SetHeader(NormalHeader);
$html->SetMenu(NormalMenu);
$html->frame();
$new = new edit();
if(isset($_POST['title'])) {
          $new->regist();
          $disp = new disp();
          $disp->edit("shosai.php");
} elseif (isset($_POST['edit'])) {
          $edit = new disp();
          $edit->edit("shosai.php");
} else {
          $html->new_page();
}
$html->HtmlEnd();
$database->close();
?>
```

【list.php】 記事一覧ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp();
$html->HtmlStart(記事一覧);
$html->SetHeader(NormalHeader);
$html->SetMenu(NormalMenu);
$html->frame();
$disp = new disp();
$disp->ichiran();
$html->HtmlEnd();
$database->close();
?>
```

【shosai.php】 記事詳細表示ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
if(isset($_POST['honbun'])){ //登録ボタンが押されたとき
          $regist = new edit();
          $regist->update();
```

```
}else if(isset($_POST['title'])){      //新規ページから登録された時
          $write = new edit();
          $write->regist();
} else if ( isset($_POST['CommentHonbun'])) {
          $CommentRegist = new edit();
          $CommentRegist->CommentRegist();
}
$html = new disp;
$html->HtmlStart("詳細");
$html->SetHeader(NormalHeader);
$html->SetMenu(NormalMenu);
$html->frame();
if (isset($_POST['prev'])){  // プレビューボタンが押されたとき
          $prev = new disp;
          $prev->prev();
} else {
          $disp = new disp();
          $disp->shosai_disp();
}
$html->HtmlEnd();
$database->close();
?>
```

【search.php】 検索結果の表示ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp();
$html->HtmlStart(検索結果);
$html->SetHeader(NormalHeader);
$html->SetMenu(NormalMenu);
$html->frame();
$disp = new disp();
$disp->Search();
$html->HtmlEnd();
$database->close();
?>
```

【method.php】 メソッド一覧ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp();
$html->HtmlStart(メソッド);
$html->SetHeader(NormalHeader);
$html->SetMenu(NormalMenu);
$html->frame();
$disp = new disp();
$disp->method_shosai();
$database->close();
$html->HtmlEnd();
?>
```

【Setup.php】 管理設定画面

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp;
$html->HtmlStart(設定メニュー);
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
$disp = new disp();
$disp->setupTop();
$html->HtmlEnd();
$database->close();
?>
```

【SetupList.php】 管理設定画面 記事一覧

```
<?php
require_once "system/system.php";
$database = new database;
$database->setDatabase(wiki_oop_tani);
$database->connect();
if ( isset($_POST['topID'])){
          $write = new edit();
          $write->RemoveData();
}
$html = new disp();
$html->HtmlStart(設定_記事一覧);
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
if ( isset($_POST['id'])){
          $delete = new edit();
          $delete->del();
}
$disp = new disp();
$disp->SetupIchiran();
$html->HtmlEnd();
$database->close();
?>
```

【SetupShosai.php】 管理設定画面 記事詳細

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp;
$html->HtmlStart("設定_詳細");
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
if (isset($_POST['prev'])){  // プレビューボタンが押されたとき
          $prev = new disp;
          $prev->prev();
} elseif(isset($_POST['title'])) {
```

```
          $regist = new edit;
          $regist->update();
          $regist->dispSetupContents();
} else {
          $disp = new disp();
          $disp->SetupEdit("SetupShosai.php");
}
$html->HtmlEnd();
$database->close();
?>
```

【SetupEdit.php】 管理設定画面 記事編集ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp;
$html->HtmlStart(設定_記事編集);
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
$regist = new edit();
if(isset($_POST['TopPage'])){   //トップページに登録したい時
          $write = new edit();
          $write->RemoveData();
} else {
          $disp = new disp();
          $disp->edit();
}
$html->HtmlEnd();
$database->close();
?>
```

【SetupComList.php】 管理設定画面 コメント一覧

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp();
$html->HtmlStart(設定_コメント一覧);
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
if ( isset($_POST['id'])) {
          $del = new edit;
          $del->CommentDel();
}
$disp = new disp();
$disp->CommentList();
$html->HtmlEnd();
$database->close();
?>
```

【SetupComShosai.php】 管理設定画面 コメント詳細

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
$html = new disp;
$html->HtmlStart("設定_コメント詳細");
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
$disp = new disp();
$disp->CommentShosai();
$html->HtmlEnd();
$database->close();
?>
```

【SetupWiki.php】 管理設定画面 諸設定ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
if(isset($_POST['WikiTitle'])){
          $disp = new edit();
          $disp->RemoveData();
}
$html = new disp();
$html->HtmlStart(詳細設定);
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
$disp = new disp();
$disp->setupList();
$html->HtmlEnd();
$database->close();
?>
```

【SetupCss.php】 管理設定画面 スタイル設定ページ

```
<?php
require_once "system/system.php";
$database = new database;
$database->connect();
if(isset($_POST['color'])){
          $disp = new edit();
          $disp->UpdateCss();
}
$html = new disp();
$html->HtmlStart(CSS 設定);
$html->SetHeader(SetupHeader);
$html->SetMenu(SetupMenu);
$html->frame();
if (isset($_POST['reset'])) {
          $reset = new edit();
          $reset->setCssData(system::StyleSheetC);
          $reset->EditCss();
} else {
          $edit = new disp();
```

```
            $edit->setCssData(system::StyleSheet);
            $edit->EditCss();
}
$html->HtmlEnd();
$database->close();
?>
```