PHP TRAINING MODUL 3: Membuat program login, registrasi dan update data user

Materi

Pada pertemuan ke-3 ini kita akan mebahas materi berikut ini:

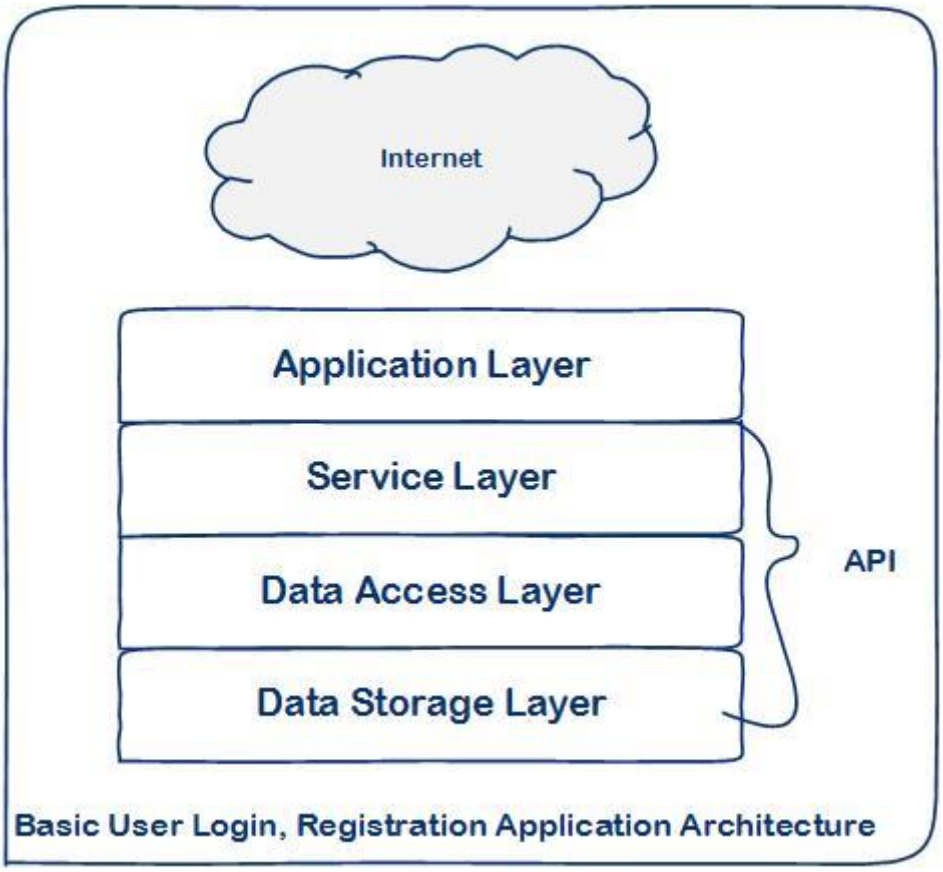
- Application architecture
- Designing the API
- User registration
- User login and logout
- User profile view and update

Sebagian aplikasi web membutuhkan pengamanan, pengaturan dan pembatasan terhadap sumber daya yang dimiliki oleh web site tersebut. Untuk melakukan itu semua diperlukan mekanisme login, manajemen user dan manajemen hak akses. Berikut ini beberapa contoh kasus yang membutuhkan pengaturan akses:

- Administrasi E-commerce
- Konfedensial
- Akses Pembayaran

Memahami Arsitektur Aplikasi

Arsitektur dibutuhkan dalam rangka membangun lapisan-lapisan dalam penyimpanan data, pengaksesan data, aplikasi layanan dan aplikasi. Berikut ini adalah rancangan arsitektur aplikasi yang akan kita bangun dalam training pertemuan ke-3 ini:



Untuk setiap layer dapat ditunjukan dalam sekelompok dari analogi proses secara logis, layer data storage bertugas seperti halnya sebuah sumber data, seperti sebuah database relasional, filesystem, atau tipe sumber data lainya. Layer data access berkomunikasi degnan sumber data untuk membaca atau menulis data ke layer data storage, dan menyediakan sebuah abstraksi yang baik dalam sumber data untuk dikirimkan ke layer service. Layer service adalah sebuah layer tengah dari data persistence dan layer aplikasi, dan juga menyediakan layanan lain. Layer aplikasi adalah layer tertinggi yang berhubungan langsung dengan user melalui halaman-halaman web.

Rancangan User Interface

Berikut ini adalah rancangan user interface untuk aplikasi web login dan maintenence user:

Registrasi

User Registration

Name:

Email:

Password:

Phone:

submit

Already registered? Login here

Login

User Login

Email:

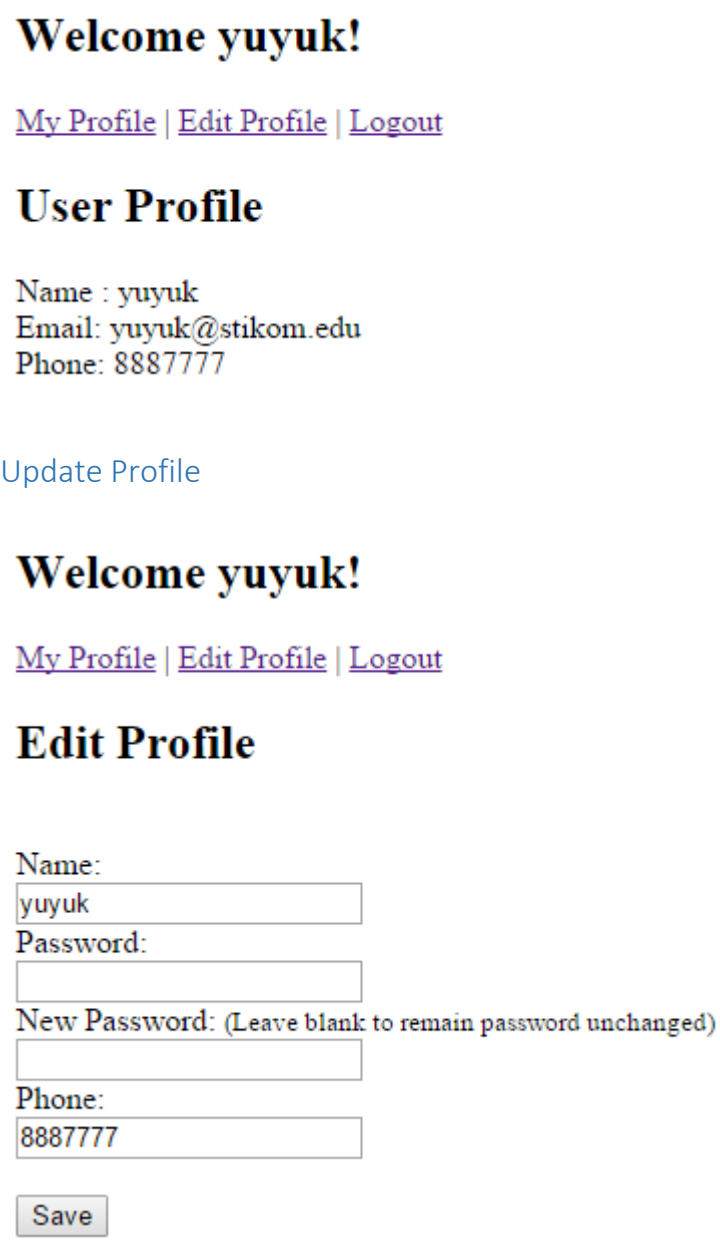
Password:

Remember me next time

Login

New User? Register here

View Profile



Update Profile

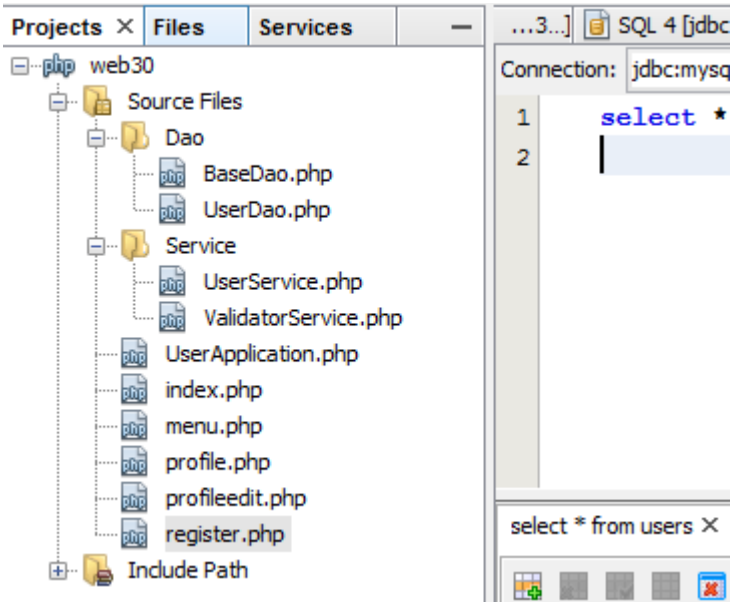
Merancang Database

Untuk merancang dan membuat database kita bisa melihat kembali langkah-langkanya yang sudah kita pelajari pada modul sebelumya. Buatlah dabase MySQL dengan nama dbuser. Kemudian tambahkan/buat tabel users menggunakan perintah SQL seperti berikut:

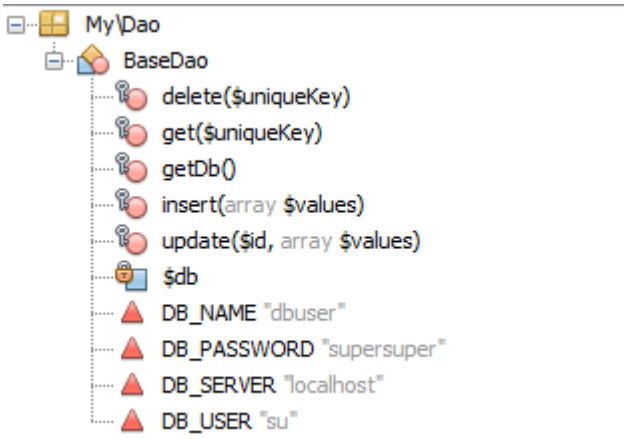
```
4
5  CREATE TABLE `users` (
6    `id` bigint(20) NOT NULL AUTO_INCREMENT,
7    `useremail` varchar(50) NOT NULL,
8    `password` char(32) NOT NULL,
9    `userhash` char(32) NOT NULL,
10   `userlevel` tinyint(4) NOT NULL,
11   `username` varchar(100) NOT NULL,
12   `phone` varchar(20) NULL,
13   `timestamp` int(11) unsigned NOT NULL,
14   PRIMARY KEY (`id`),
15   UNIQUE KEY `useremail` (`useremail`)
16 ) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

Lab 1: Membuat Project PHP Application: Data Storage Layer

1. Buat sebuah project PHP Application dengan nama **web30**
2. Tambahkan folder berikut ini
    a. Dao, folder ini berisi class-class pada layer data storage
    b. Service, folder ini berisi class pada layer Service

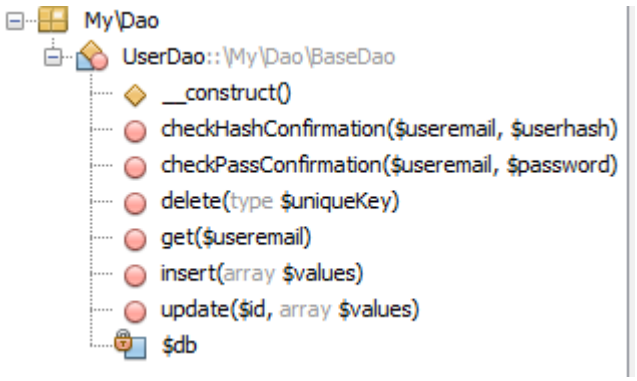


3. Tambahkan sebuah PHP Class dengan nama BaseDao dengan struktur class seperti berikut:




```
<?php
namespace My\Dao;
abstract class BaseDao {
    private $db          = null;

    const DB_SERVER     = "localhost";
    const DB_USER       = "root";
    const DB_PASSWORD   = "supersuper";
    const DB_NAME       = "dbuser";

    protected final function getDb(){
        $dsn =
'mysql:dbname='.self::DB_NAME.';host='.self::DB_SERVER;

        try {
            $this->db = new \PDO($dsn, self::DB_USER,
self::DB_PASSWORD);
        } catch (PDOException $e) {
            throw new \Exception('Connection failed: ' . $e-
>getMessage());
        }

        return $this->db;
    }

    abstract protected function get($uniqueKey);
    abstract protected function insert(array $values);
    abstract protected function update($id, array $values);
    abstract protected function delete($uniqueKey);
}
?>
```

4. Tambahkan PHP Class dengan nama UserDao yang merupakan turunan dari class BaseDao, struktur class User Dao adalah sebagai berikut:




```
<?php

namespace My\Dao;

class UserDao extends BaseDao {
private $db = null;
public function __construct() {
$this->db = $this->getDb();
}
}
$userDao = new \My\Dao\UserDao;
?>
```

5. Masih di dalam class UserDao, tuliskan implementasi dari method get() sebagai berikut:

```
public function get($useremail) {
        $statement = $this->db->prepare("SELECT * FROM users
WHERE useremail = :useremail LIMIT 1 ");
        $statement->bindParam(':useremail', $useremail);
        $statement->execute();

        if ($statement->rowCount() > 0) {
            return $statement->fetch();
        }

        return false;
    }
```

6. Tambahkan method insert() untuk menambah data user baru:

```
public function insert(array $values) {
$sql = "INSERT INTO users ";
$fields = array_keys($values);
$vals = array_values($values);
$sql .= '('.implode(',', $fields).') ';
$arr = array();
foreach ($fields as $f) {
$arr[] = '?';
}
$sql .= 'VALUES ('.implode(',', $arr).') ';
```

```
$statement = $this->db->prepare($sql);
foreach ($vals as $i=>$v) {
$statement->bindValue($i+1, $v);
}
return $statement->execute();
}
```

7. Tambahkan method update() untuk melakukan update data sebuah user

```
public function update($id, array $values) {
        $sql = "UPDATE users SET ";
        $fields = array_keys($values);
        $vals = array_values($values);

        foreach ($fields as $i=>$f) {
            $fields[$i] .= ' = ? ';
        }

        $sql .= implode(',', $fields);
        $sql .= " WHERE id = " . (int)$id ." LIMIT 1 ";

        $statement = $this->db->prepare($sql);
        foreach ($vals as $i=>$v) {
            $statement->bindValue($i+1, $v);
        }

        $statement->execute();
    }
```

8. Karena keterbatasan waktu untuk sementara method delete dikosongi dulu, tetapi tetap didefinisikan ulang dalam class UserDao

```
public function delete($uniqueKey) {
}
```

9. Pada saat penambahan user baru biasanya perlu dilakakukan pengujian terhadap alamat email yang digunakan apakah sudah ada dalah tabel users atau belum, untuk itu perlu kita tambahkan sebuah fungsi yang digunakan untuk menghitung berapa jumlah alamat email yang dimasukkan melalui parameter method.

```
public function useremailTaken($useremail) {
        $statement = $this->db->prepare("SELECT id FROM users
WHERE useremail = :useremail LIMIT 1 ");
        $statement->bindParam(':useremail', $useremail);
        $statement->execute();

        return ($statement->rowCount() > 0 );
    }
```

10. Untuk melakukan konfirmasi username dan password pada saat login maka kita tambahkan method **checkPassConfirmation**() sebagai berikut:

```
public function checkPassConfirmation($useremail, $password)
{

        $statement = $this->db->prepare("SELECT password FROM
users WHERE useremail = :useremail LIMIT 1 ");
        $statement->bindParam(':useremail', $useremail);
        $statement->execute();

        if ($statement->rowCount() > 0) {
            $row = $statement->fetch();

            return ($password == $row['password']);
        }

        return false;
    }
```

11. Method terakhir pada class UserDao adalah untuk melakukan pengecekan userhast yang berisi angka unik untuk setiap session.

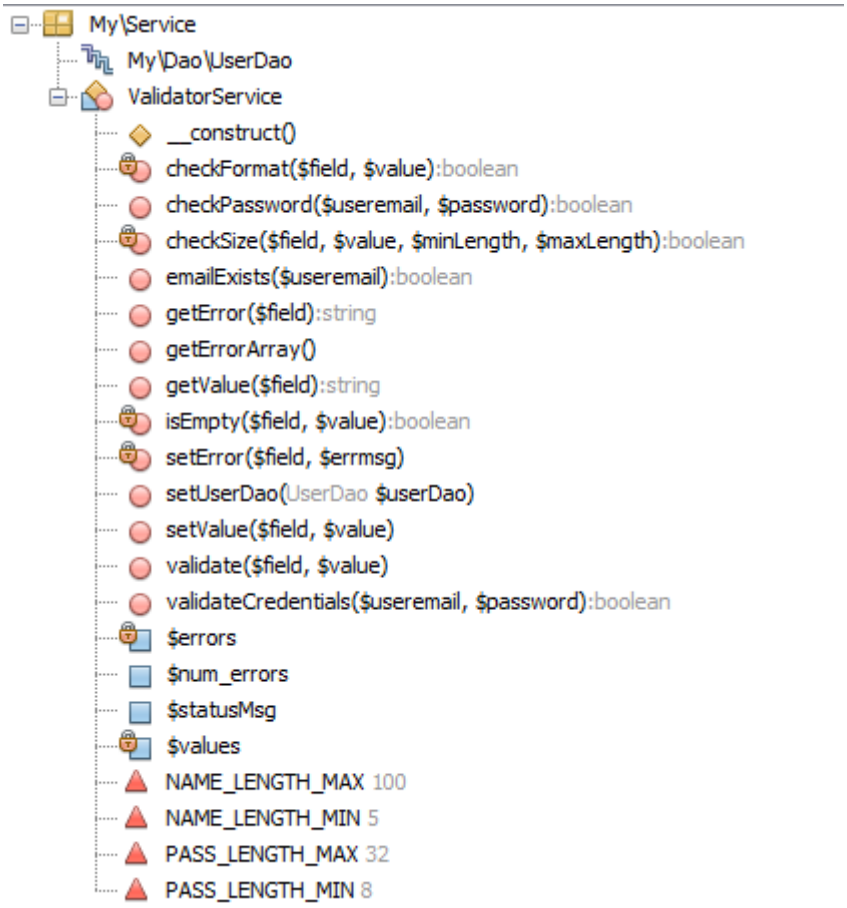
```
public function checkHashConfirmation($useremail, $userhash)
{

        $statement = $this->db->prepare("SELECT userhash FROM
users WHERE useremail = :useremail LIMIT 1");
        $statement->bindParam(':useremail', $useremail);
        $statement->execute();

        if ($statement->rowCount() > 0) {
            $row = $statement->fetch();

            return ($userhash == $row['userhash']);
        }

        return false;
    }
```

12. Pada bagian akhir class (setelah kurung kurawal class) tambahkan sebuah variable yang merupakan instance dari class UserDao

```
$userDao = new \My\Dao\UserDao;
```

Lab 2: Membuat Access Layer

1. Tambahkan PHP Class di dalam folder Service dengan nama ValidatorService yang memilki struktur class sebagai berikut:



2. Tambahkan beberapa konstatnta serperti berikut:

```
<?php
namespace My\Service;
use My\Dao\UserDao;

class ValidatorService {
private $values = array();
private $errors = array();
public $statusMsg = null;
public $num_errors;
const NAME_LENGTH_MIN = 5;
const NAME_LENGTH_MAX = 100;
const PASS_LENGTH_MIN = 8;
const PASS_LENGTH_MAX = 32;
public function __construct() {
}
public function setUserDao(UserDao $userDao){
$this->userDao = $userDao;
}
}
```

```
$validator = new \My\Service\ValidatorService;
$validator->setUserDao($userDao);
?>
```

3. Baiklah sekarang mari kita mulai membuat constructor dari class ValidatorService

```
public function __construct() {

        if (isset($_SESSION['value_array']) &&
isset($_SESSION['error_array'])) {
            $this->values = $_SESSION['value_array'];
            $this->errors = $_SESSION['error_array'];
            $this->num_errors = count($this->errors);

            unset($_SESSION['value_array']);
            unset($_SESSION['error_array']);
        } else {
            $this->num_errors = 0;
        }

        if (isset($_SESSION['statusMsg'])) {
            $this->statusMsg = $_SESSION['statusMsg'];
            unset($_SESSION['statusMsg']);
        }
    }
```

4. Tambahkan beberapa method berikut:

```
    public function setUserDao(UserDao $userDao){
        $this->userDao = $userDao;
    }

    public function setValue($field, $value) {
        $this->values[$field] = $value;
    }

    public function getValue($field) {
        if (array_key_exists($field, $this->values)) {
            return htmlspecialchars(stripslashes($this-
>values[$field]));
        } else {
            return "";
        }
    }

    private function setError($field, $errmsg) {
        $this->errors[$field] = $errmsg;
        $this->num_errors = count($this->errors);
    }

    public function getError($field) {
        if (array_key_exists($field, $this->errors)) {
```

```
            return $this->errors[$field];
        } else {
            return "";
        }
    }

    public function getErrorArray() {
        return $this->errors;
    }
```

5. Tuliskan method untuk melakukan validasi seperti berikut:

```
public function validate($field, $value) {
        $valid = false;

        if ($valid == $this->isEmpty($field, $value)) {

            $valid = true;
            if ($field == "name")
                $valid = $this->checkSize($field, $value,
self::NAME_LENGTH_MIN, self::NAME_LENGTH_MAX);

            if ($field == "password" || $field ==
"newpassword")
                $valid = $this->checkSize($field, $value,
self::PASS_LENGTH_MIN, self::PASS_LENGTH_MAX);

            if ($valid)
                $valid = $this->checkFormat($field, $value);
        }

        return $valid;
    }
```

6. Tuliskan method untuk melakukan validasi format masukan:

```
private function isEmpty($field, $value) {
        $value = trim($value);
        if (empty($value)) {
            $this->setError($field, "Field value not
entered");
            return true;
        }

        return false;
    }

    private function checkFormat($field, $value) {

        switch ($field) {
            case 'useremail':
                $regex = "/^[_+a-z0-9-]+(\.[_+a-z0-9-]+)*"
                        . "@[a-z0-9-]+(\.[a-z0-9-]{1,})*"
```

```
                        . "\.([a-z]{2,}){1}$/i";
                $msg = "Email address invalid";
                break;
            case 'password':
            case 'newpassword':
                $regex = "/^([0-9a-z])+$/i";
                $msg = "Password not alphanumeric";
                break;
            case 'name':
                $regex = "/^([a-z ])+$/i";
                $msg = "Name must be alphabetic";
                break;
            case 'phone':
                $regex = "/^([0-9])+$/";
                $msg = "Phone not numeric";
                break;
            default:;
        }

        if (!preg_match($regex, ( $value = trim($value)))) {
            $this->setError($field, $msg);
            return false;
        }

        return true;
    }

    private function checkSize($field, $value, $minLength,
$maxLength) {
        $value = trim($value);

        if (strlen($value) < $minLength || strlen($value) >
$maxLength) {
            $this->setError($field, "Value length should be
within ".$minLength." & ".$maxLength." characters");
            return false;
        }

        return true;
    }
```

7. Tambahkan method validateCredentials() dan emailExists()

```
public function validateCredentials($useremail, $password) {

        $result = $this->userDao-
>checkPassConfirmation($useremail, md5($password));

        if ($result === false) {
            $this->setError("password", "Email address or
password is incorrect");
            return false;
        }
```

```
        return true;
    }

public function emailExists($useremail) {

        if ($this->userDao->useremailTaken($useremail)) {
            $this->setError('useremail', "Email already in
use");
            return true;
        }

        return false;
    }
```

8. Tambahkan satu method lagi yang digunakan untuk melakukan proses login

```
public function checkPassword($useremail, $password) {

        $result = $this->userDao-
>checkPassConfirmation($useremail, md5($password));

        if ($result === false) {
            $this->setError("password", "Current password
incorrect");
            return false;
        }

        return true;
    }
```

Membuat class UserService

1. Tambahkan sebuah PHP Class dalam folder Service dengan nama UserService.
2. Tambahkan variable dan konstansta seperti berikut:

```
<?PHP
namespace My\Service;
use My\Dao\UserDao;
use My\Service\ValidatorService;

class UserService {

    public $useremail;
    private $userid;
    public $username;
    public $userphone;
    private $userhash;
    private $userlevel;
    public $logged_in;

    const ADMIN_EMAIL = "admin@mysite.com";
    const GUEST_NAME  =  "Guest";
```

```
    const ADMIN_LEVEL = 9;
    const USER_LEVEL  =  1;
    const GUEST_LEVEL = 0;

    const COOKIE_EXPIRE =  8640000;  //60*60*24*100 seconds =
100 days by default
    const COOKIE_PATH = "/";  //Available in whole domain
```

3. Tambahkan constuctor seperti berikut:

```
public function __construct(UserDao $userDao,
ValidatorService $validator) {

        $this->userDao = $userDao;
        $this->validator = $validator;

        $this->logged_in = $this->isLogin();

        if (!$this->logged_in) {
            $this->useremail = $_SESSION['useremail'] =
self::GUEST_NAME;
            $this->userlevel = self::GUEST_LEVEL;
        }
    }
```

4. Selanjutnya kita perlu membuat method isLogin() untuk mengetahui status login user:

```
private function isLogin() {

        if (isset($_SESSION['useremail']) &&
isset($_SESSION['userhash']) &&
                $_SESSION['useremail'] != self::GUEST_NAME) {

            if ($this->userDao-
>checkHashConfirmation($_SESSION['useremail'],
$_SESSION['userhash']) === false) {
                unset($_SESSION['useremail']);
                unset($_SESSION['userhash']);
                unset($_SESSION['userid']);
                return false;
            }

            $userinfo = $this->userDao-
>get($_SESSION['useremail']);
            if(!$userinfo){
                return false;
            }

            $this->useremail = $userinfo['useremail'];
            $this->userid = $userinfo['id'];
            $this->userhash = $userinfo['userhash'];
            $this->userlevel = $userinfo['userlevel'];
            $this->username = $userinfo['username'];
```

```
            $this->userphone = $userinfo['phone'];
            return true;

        }

        if (isset($_COOKIE['cookname']) &&
isset($_COOKIE['cookid'])) {
            $this->useremail = $_SESSION['useremail'] =
$_COOKIE['cookname'];
            $this->userhash = $_SESSION['userhash'] =
$_COOKIE['cookid'];
            return true;
        }

        return false;
    }
```

5. Berikut ini adalah method login() yang digunakan untuk melakukan proses login

```
public function login($values) {

        $useremail = $values['useremail'];
        $password = $values['password'];
        $rememberme = isset($values['rememberme']);

        $this->validator->validate("useremail", $useremail);
        $this->validator->validate("password", $password);

        if ($this->validator->num_errors > 0) {
            return false;
        }

        if (!$this->validator-
>validateCredentials($useremail, $password)) {
            return false;
        }


        $userinfo = $this->userDao->get($useremail);
        if(!$userinfo){
            return false;
        }

        $this->useremail = $_SESSION['useremail'] =
$userinfo['useremail'];
        $this->userid = $_SESSION['userid'] =
$userinfo['id'];
        $this->userhash = $_SESSION['userhash'] =
md5(microtime());
        $this->userlevel = $userinfo['userlevel'];
        $this->username = $userinfo['username'];
        $this->userphone = $userinfo['phone'];
```

```
        $this->userDao->update($this->userid,
array("userhash" => $this->userhash));


        if ($rememberme == 'true') {
            setcookie("cookname", $this->useremail, time() +
self::COOKIE_EXPIRE, self::COOKIE_PATH);
            setcookie("cookid", $this->userhash, time() +
self::COOKIE_EXPIRE, self::COOKIE_PATH);
        }

        return true;
    }
```

6. Tambahkan method register untuk melakukan proses registrasi user baru:

```
public function register($values) {
        $username = $values['name'];
        $useremail = $values['useremail'];
        $password = $values['password'];
        $phone = $values['phone'];
             //echo $useremail . "------";
        $this->validator->validate("name", $username);
        $this->validator->validate("useremail", $useremail);
        $this->validator->validate("password", $password);
        $this->validator->validate("phone", $phone);
        //echo "err: " . $this->validator->num_errors;
        if ($this->validator->num_errors > 0) {
            return false;
        }

        if($this->validator->emailExists($useremail)) {
            return false;
        }

        $ulevel = (strcasecmp($useremail, self::ADMIN_EMAIL)
== 0) ? self::ADMIN_LEVEL : self::USER_LEVEL;

        return $this->userDao->insert(array(
            'useremail' => $useremail, 'password' =>
md5($password),"userhash"=> 0,
            'userlevel' => $ulevel, 'username' => $username,
            'phone' => $phone, 'timestamp' => time()
            ));

    }
```

7. Tambahkan method getUser() untuk menampilkan data user berdasarkan alamat emailnya:

```
public function getUser($useremail){

        $this->validator->validate("useremail", $useremail);
```

```
        if ($this->validator->num_errors > 0) {
            return false;
        }
        if (!$this->validator->emailExists($useremail)) {
            return false;
        }

        $userinfo = $this->userDao->get($useremail);

        if($userinfo){
            return $userinfo;
        }

        return false;
    }
```

8. Tambahkan method untuk melakukan proses update data user:

```
public function update($values) {
        $username = $values['name'];
        $phone = $values['phone'];
        $password = $values['password'];
        $newPassword = $values['newpassword'];

        $updates = array();

        if($username) {
            $this->validator->validate("name", $username);
            $updates['username'] = $username;
        }

        if($phone) {
            $this->validator->validate("phone", $phone);
            $updates['phone'] = $phone;
        }

        if($password && $newPassword){
            $this->validator->validate("password",
$password);
            $this->validator->validate("newpassword",
$newPassword);
        }

        if ($this->validator->num_errors > 0) {
            return false;
        }

        if($password && $newPassword){
            if ($this->validator->checkPassword($this-
>useremail, $password)===false) {
                return false;
            }

            $updates['password'] = md5($newPassword);
```

```
        }

        $this->userDao->update($this->userid, $updates);

        return true;
    }
```

9. Tambahkan method logout()

```
public function logout() {

        if (isset($_COOKIE['cookname']) &&
isset($_COOKIE['cookid'])) {
            setcookie("cookname", "", time() -
self::COOKIE_EXPIRE, self::COOKIE_PATH);
            setcookie("cookid", "", time() -
self::COOKIE_EXPIRE, self::COOKIE_PATH);
        }

        unset($_SESSION['useremail']);
        unset($_SESSION['userhash']);

        $this->logged_in = false;

        $this->useremail = self::GUEST_NAME;
        $this->userlevel = self::GUEST_LEVEL;
    }
```

10. Tambahkan mehtod untuk mengechek apakah user tersebut memilki hak akses Admin atau bukan.

```
public function isAdmin() {
        return ($this->userlevel == self::ADMIN_LEVEL ||
                $this->useremail == self::ADMIN_EMAIL);
    }
```

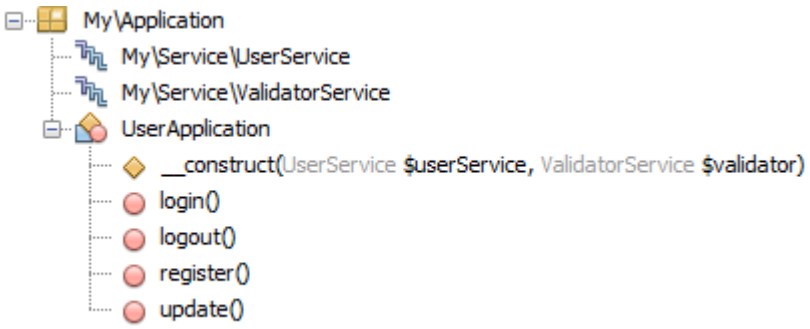
11. Tambahkan sebuah variable di bagian paling bawah file UserService.php dengan nama $userService:

```
$userService = new \My\Service\UserService($userDao,
$validator);
```

Lab 3: Membuat Application Layer

Membuat Class UserApplication

1. Tambahkan sebuah PHP Class pada root directory project web30 dengan nama UserApplication dengan struktur class sebagai berikut:




```
<?php
namespace My\Application;
use My\Service\UserService;
use My\Service\ValidatorService;
session_start();

/* include the APIs */
require_once "Dao/BaseDao.php";
require_once "Dao/UserDao.php";
require_once "Service/ValidatorService.php";
require_once "Service/UserService.php";
class UserApplication {

}

$userApp = new
\My\Application\UserApplication($userService, $validator);
```

2. Kita mulai dengan membuat method __construct() yang merupakan konstruktor dari class UserApplication:

```
public function __construct(UserService $userService,
ValidatorService $validator) {

        $this->userService = $userService;
        $this->validator = $validator;

        if (isset($_POST['login'])) {

            $this->login();
        }
        else if (isset($_POST['register'])) {

            $this->register();
        }
        else if (isset($_POST['update'])) {

            $this->update();
        }
        else if ( isset($_GET['logout']) ) {

            $this->logout();
```

```
        }
    }
```

3. Tambahkan method login() :

```
public function login() {

        $success = $this->userService->login($_POST);

        if ($success) {
            $_SESSION['statusMsg'] = "Successful login!";
        } else {
            $_SESSION['value_array'] = $_POST;
            $_SESSION['error_array'] = $this->validator-
>getErrorArray();
        }
        header("Location: index.php");
    }
```

4. Tambahkan method register():

```
public function register() {

        $success = $this->userService->register($_POST);
           echo "<h2>" . $success . $_POST['name'] .
"</h2>";

        if ($success) {
            $_SESSION['statusMsg'] = "Registration was
successful!";
            header("Location: index.php");
        } else {
            $_SESSION['value_array'] = $_POST;
            $_SESSION['error_array'] = $this->validator-
>getErrorArray();
            header("Location: register.php");
        }

    }
```

5. Tambahkan method update():

```
public function update() {

        $success = $this->userService->update($_POST);


        if ($success) {
            $_SESSION['statusMsg'] = "Successfully
Updated!";
            header("Location: profile.php");
```

```
        } else {
            $_SESSION['value_array'] = $_POST;
            $_SESSION['error_array'] = $this->validator-
>getErrorArray();
            header("Location: profile-edit.php");
        }
    }
```

6. Tambahkan method logout()

```
public function logout(){

        $success = $this->userService->logout();
        header("Location: index.php");
    }
```

Membuat User Interface

1. Membuat/memodifikasi index.php

```
<?php
require_once 'UserApplication.php';
?>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<title></title>
</head>
<body>
</body>
</html>
```

2. Modifikasi kode program diantara tag <body> .. </body>

```
<?php
        include 'menu.php';

        if (!$userService->logged_in) {
        ?>

            <h2>User Login</h2>
            <br />
            <?php

            if ($validator->num_errors > 0) {
                echo "<span style=\"color:#ff0000;\">"
. $validator->num_errors . " error(s) found</span>";
            }
            ?>
```

```
            <form action="UserApplication.php"
method="POST">
                Email: <br />
                <input type="text" name="useremail"
value="<?= $validator->getValue("useremail") ?>"/>
                    <?php echo "<span
style=\"color:#ff0000;\">".$validator-
>getError("useremail")."</span>"; ?>
                <br />
                Password:<br />
                <input type="password" name="password"
value=""/>
                    <?php echo "<span
style=\"color:#ff0000;\">".$validator-
>getError("password")."</span>"; ?>
                <br />
                <input type="checkbox"
name="rememberme" <?=($validator-
>getValue("rememberme") != "")?"checked":""?>/>
                <font size="2">Remember me next time
</font>
                <br />
                <input type="hidden" name="login"
value="1">
                <input type="submit" value="Login">
            </form>
            <br />
            New User? <a href="register.php">Register
here</a>
        <?php
        }
        ?>
```

3. Tambahkan sebuah file PHP dengan nama register.php, tuliskan kode berikut:

```
<?php
require_once 'UserApplication.php';
?>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <?php

        include 'menu.php';

        if (!$userService->logged_in) {
            ?>
```

```
            <h2>User Registration</h2><br />
            <?php

            if ($validator->num_errors > 0) {
                echo "<span style=\"color:#ff0000;\">"
. $validator->num_errors . " error(s) found</span>";
            }
            ?>

            <form id="frm1"
action="UserApplication.php" method="POST">
                Name: <br />
                <input type="text" name="name"
value="<?= $validator->getValue("name") ?>"/> <?php
echo "<span style=\"color:#ff0000;\">".$validator-
>getError("name")."</span>";
                    ?>
                <br />
                Email: <br />
                <input type="text" name="useremail"
value="<?= $validator->getValue("useremail") ?>"/>
                    <?php echo "<span
style=\"color:#ff0000;\">".$validator-
>getError("useremail")."</span>"; ?>
                <br />
                Password:<br />
                <input type="password" name="password"
value=""/>
                    <?php echo "<span
style=\"color:#ff0000;\">".$validator-
>getError("password")."</span>"; ?>
                <br />
                Phone: <br />
                <input type="text" name="phone"
value="<?= $validator->getValue("phone") ?>"/>
                    <?php echo "<span
style=\"color:#ff0000;\">".$validator-
>getError("phone")."</span>"; ?>
                <br /><br />
                <input type="hidden" name="register"
value="1"/>
                <input type="submit" value="Register"/>
            </form>
            <br />
            Already registered? <a
href="index.php">Login here</a>
        <?php
        }
        ?>
    </body>
</html>
```

4. Tambahkan file profile.php untuk menampilkan profile dari user yang sedang aktif

```
<?php
require_once 'UserApplication.php';
?>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        include 'menu.php';

        if ($userService->logged_in) {
            echo '<h2>User Profile</h2>';

            echo "Name : " . $userService->username .
"<br />";
            echo "Email: " . $userService->useremail .
"<br />";
            echo "Phone: " . $userService->userphone .
"<br />";
        }
        ?>
    </body>
</html>
```

5. Tambahkan file profile-edit.php:

```
<?php
require_once 'UserApplication.php';
?>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">
        <title></title>
    </head>
    <body>
        <?php
        include 'menu.php';

        if ($userService->logged_in) {
            ?>

            <h2>Edit Profile</h2><br />

            <?php

            if ($validator->num_errors > 0) {
                echo "<span style=\"color:#ff0000;\">"
. $validator->num_errors . " error(s) found</span>";
```

```
            }
            ?>

            <form action="UserApplication.php"
method="POST">
                Name: <br />
                <input type="text" name="name"
value="<?= ($validator->getValue("name") != "") ?
$validator->getValue("name") : $userService->username
?>"/>
                    <?php echo "<span
style=\"color:#ff0000;\">" . $validator-
>getError("name") . "</span>"; ?>
                <br />
                Password:<br />
                <input type="password" name="password"
value=""/>
                    <?php echo "<span
style=\"color:#ff0000;\">" . $validator-
>getError("password") . "</span>"; ?>
                <br />
                New Password: <font size="2">(Leave
blank to remain password unchanged)</font><br />
                <input type="password"
name="newpassword" value=""/>
                    <?php echo "<span
style=\"color:#ff0000;\">" . $validator-
>getError("newpassword") . "</span>"; ?>
                <br />
                Phone: <br />
                <input type="text" name="phone"
value="<?= ($validator->getValue("phone") != "") ?
$validator->getValue("phone") : $userService->userphone
?>"/>
                    <?php echo "<span
style=\"color:#ff0000;\">" . $validator-
>getError("phone") . "</span>"; ?>
                <br /><br />
                <input type="hidden" name="update"
value="1">
                <input type="submit" value="Save">
            </form>
            <?php
        }
        ?>
    </body>
</html>
```

6. Tambahkan file menu.php yang digunakan untuk menampilkan menu jika user sudah melakukan login

```
<?php
```

```
if (isset($validator->statusMsg)) {
    echo "<span style=\"color:#207b00;\">" .
$validator->statusMsg . "</span>";
}

if ($userService->logged_in) {
    echo "<h2>Welcome $userService->username!</h2>";
    echo "<a href='profile.php'>My Profile</a> | "
    . "<a href='profile-edit.php'>Edit Profile</a> | "
    . "<a
href='UserApplication.php?logout=1'>Logout</a> ";
}
?>
```