Labb rapport 2

Internet Applications, ID1354

Evan Saboo
saboo@kth.se

2015-10-28

1 Introduktion

Målet med andra labben är att få en grundläggande kunskap om PHP språket och man ska kunna använda PHP till serverprogrammering.

I den första uppgiften ska man skapa en PHP baserat inloggningsfunktion som ska implementeras i hemsidan från den första labben. Den andra uppgiften går ut på att skapa en PHP baserat kommentars sektion vars funktion ska fungera med inloggningsfunktionen. I den sista deluppgiften ska man implementera en ”Edit” funktion i kommentar sektionen som ändrar sin egen kommentar.
Den första valfria uppgiften går ut på att skapa en registreringssida som låter nya användare registrera sig i hemsidan.

Jag samarbetade med David Nartey (dnartey@kth.se) med uppgifterna.

2 Litteratur Studie

PHP funktionerna utfördes med hjälp av dessa källor:
* De tre föreläsningarna som handlade om PHP hjälpte mig att förstå hur PHP språket fungerar i grunden och hur man använder det för att skapa olika funktioner.

* Vi fick mycket hjälp i handledningarna med vår inloggningsfunktion som vi hade fastnat på.

* http://www.w3schools.com/php/php_ref_array.asp - Hjälpmedel för att omvandla data från MySQL databasen till en Array.

* http://www.w3schools.com/html/html_forms.asp Hjälpmedel för att skapa inloggning och kommentar formulär.

* I boken ’Programming the WWW 6th av Sebesta Robert W’ gick jag igenom några delar i kapitel 9:
- 9.1 Overview of PHP
- 9.4 Primitives, Operations, and Expressions
- 9.7 Arrays
- 9.8 Functions
- 9.13 Session Tracking

* Jag kollade också i Google och Youtube videor för att kunna fixa små och stora problem, och för att få en överblick om hur man skapar PHP funktionerna som vi behövde.

3 Metod

Den första uppgiften var inte svår att utföra och det tog oss mindre än en dag att utföra. Vi utförde uppgiften genom att skapa en enkel kommentarsformulär som använde MySQL databasen (myphpadmin) för att spara alla kommentarer i ordning. Databasen hade tre kolumner, vilket var ID, namn och kommentar. Vi använde oss av några guider för att kunna koppla PHP kommentarsfunktionen med databasen.

I den andra labbuppgiften skulle man skapa en inloggningsfunktion som fungerade med kommentarfunktionen, dvs. att man ska kunna kommentera i en sida när man var inloggad i sitt eget konto. Vi började med skapa en ny sida för inloggning och kopplade den med en recept sida för att testa inloggningsfunktionen. Vi koppade funktionen med MySQL databasen för att kunna spara alla användare och återanvända dem vid verifikationen av inloggningsförsök. Den svåraste delen med uppgiften var att gå tillbaka till den förgående sidan när inloggningsfunktionen utfördes när användaren lyckades logga in. Kursläraren Leif hjälpte oss att lösa problemet i handledningstillfällena genom att spara en länk på nuvarande sida när man tryckte på knappen ”Login”. På detta sätt kunde vi använda länken för att gå tillbaka till den förgående sidan vid lyckad inloggning.

Den tredje labbuppgiften gick ut på att implementera en ”Edit” funktion i kommentar sektionen. Funktionen ska kunna redigera en kommentar men användaren ska bara kunna redigera kommentaren han skrev. Vi hade fått inspiration från en video guide som visade hör man skapade en ”Delete” knapp som tog bort en kommentar när man tryckte på knappen, vilket hjälpte oss att skapa redigera knappen eftersom knapparnas PHP funktion var nästan likadana. En Metod kollade först om den inloggade användarens alias stämde med kommentarens alias, på detta sätt kunde vi tillåta användaren att redigare bara hans kommentarer.

Eftersom alla html filer innehöll PHP koder var vi tvungna att omvandla dem till php filer vilket ledde till hög koppling.

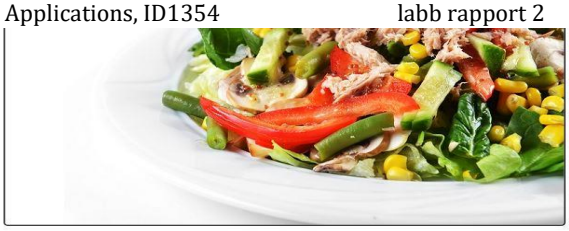
Registreringssidan utfördes i tredje seminariet eftersom vi inte hann implementera den i andra labbet. Registreringens funktioner följer MVC mönstret som är ett krav i tredje seminariet. Vi har även implementerat valideringsfunktioner t.ex. användaren kan bara registrera användarnamn som innehåller string och/eller tal. Registreringensfunktionen är kopplat med databasen för att spara nya användare istället för att spara allt i textfiler.

4 Resultat/ Diskussion

Uppgift 1. Skriva kommentarer i recept sidorna:

Denna uppgift krävde att inloggade användare hade möjlighet att skriva kommentarer.

Vi lyckades implementera uppgiften. Det var ganska svårt att hålla koll på användar-ID och deras kommentar.



- 1 tablespoon dried p
- 1 teaspoon dried dil
- 1 pinch garlic powde

Instructions

1. In a medium bowl, stir together the tuna, mayonnaise, Parmesan cheese, and onion flak
2. Season with curry powder, parsley, dill and garlic powder. Mix well and serve with crack

Comments

Comment:

comment

hello1:

Yummy!

Delete Edit

Figur 1.1 Visar användarförmåga att skriva kommentarer i ett recept sida.

* Kommentarerna och deras skapare sparas i databasen för återanvändning.

```php
<?php
// Checks if a user has logged in and includes comment box form if the statment is true.
if(isset($_SESSION['CurrentUser']))
{
include '../../php/php/commentBoxForm.php';
}

//include code (connecting to database).
include '../../php/php/databaseConnection.php';

//Takes information from MySQL database and assign it to a variable.
$result= mysql_query('SELECT * FROM comments', $connection);

//Creates a array out of the database variable and divides it into three unique variables.
while($rows = mysql_fetch_array($result))
{
  $id = $rows['id'];
  $name =$rows['name'];
  $comment = $rows['comment'];


 //Checks if current user (logged in user) is active and if current user matches the username
 // taken from the database.
  // If the statement is true assign two unique links to their own variables.
   if(isset($_SESSION['CurrentUser']) && $_SESSION['CurrentUser'] == $name )
  {
  $delink ='<li><a href="../../php/php/delete.php?id='. $id .'">Delete</a></li>';

  $EditLink ='<li><a href="../../php/php/EditForm.php?id='. $id .'">Edit</a></li>';

  //The variable 'edtiCommment' is used in the Edit form  to save the current comment use it in the edit box.
  if(isset($_GET['id']) && $_GET['id']== $id)
{
   $editComment = $comment;

}

  }
  // User is not logged in edit and delete button wonät show up in comment section.
  else{
    $delink = '';
    $EditLink = '';
  }

  //Prints out comments taken from the database and also show edit and delete button.
echo '<div id="comments"><div id="user">' .$name . ':</div><br />' . '<br />' . $comment . '<br />'
      . ' <div id="DELbutton">
    <nav>
<ul>' . $delink . '  ' . $EditLink .
            '</ul>'
            . '</nav>'
            . '</div>'
            . '<br />' . '<hr width = "auto" /></div>';

    }


      ?>
```

Figur 1.2 visar kommentar funktionen som tar kommentarer från databasen och skriver ut dem i recept sidan.
Edit knappen och delete knappen får sina funktioner när användaren har loggat in.

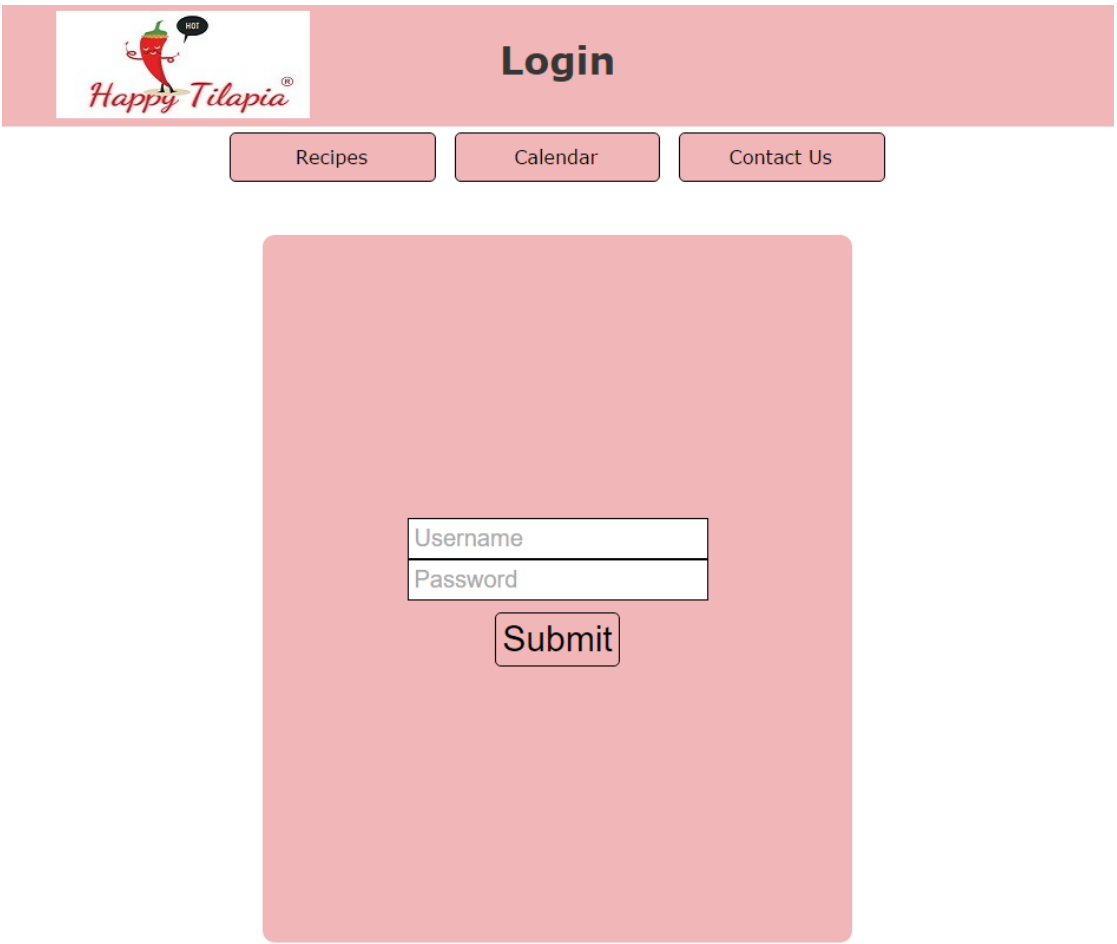
Figur 1.3 visar kommentardatabasen vilket innehåller en kommentar.

```php
<?php
//Connects to MySQL database with given name and password.
    $connection = mysql_connect("localhost","root","","happytilapia") or die(mysql_error());
     mysql_select_db("happytilapia", $connection);
    ?>
```

Figur 1.4 visar metoden för att koppla till databasen.

Uppgift 2. Validering:

Uppgiften krävde att alla användare kunde se kommentarerna men bara de inloggade användarna kunde skriva kommentarer. Uppgiften krävde också att designen för uppgifterna skulle matcha websidans design.
Nedan visas hur uppgiften utfördes:



Figur 2.1 visar in-logginsidan.

```php
<?php
//Start a session if a session is not currently active.
if(!isset($_SESSION)){
    session_start();
}

?>
<!-- login page -->
<!doctype html>
<html>
<head>
    <title>Login</title>

        <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
        <link href="../../index/mobile.css" rel="stylesheet" type="text/css" media="only screen and (min-width: 0px) and (max-width: 320px)">
        <link href="../../index/pad.css" rel="stylesheet" type="text/css" media="only screen and (min-width: 321px) and (max-width: 768px)">
        <link href="../../index/screen.css" rel="stylesheet" type="text/css" media="only screen and (min-width:769px)">


        </head>
        <body>

        <div id="container">
                    <div id="topbar">

                <!--Creating a fixed portion which adjusts to browser widow width-->
                <!--Making it a class so we can re use-->

                    <div class="fixedwidth">
                    <div id="logodiv">

                        <a href="../../index.html"><img src="../../Images/logo.png" alt="Happy Tilapia"/></a>


                    </div>
                    </div>


                <div id="topmenu">
                <h1>Login</h1>

                </div>

        </div>
</div>
    <header>
    <nav>
      <ul>
            <li><a href="../../index.html">Recipes</a>
            <ul>
                    <li><a href="../../breakfast.html">Breakfast</a></li>
                    <li><a href="../../lunch.html">Lunch</a></li>
                    <li><a href="../../dinner.html">Dinner</a></li>

            </ul>
            </li>
            <li><a href="../../calendar.html">Calendar</a></li>
        <li><a href="../../contact.html">Contact Us</a></li>

    </ul>
    </nav>
    </header>
                <div id="loginForm">
                    <?php
                    include 'loginForm.php';

                    ?>
                </div>
            </body>

            </html>
```

Figur 2.2 visar koden på login sidan.

```
<!--A simple login form used in login page-->
      <div id="loginDesign">
        <form  method="post" action="login.php">

            <input type="text" name="username" placeholder="Username">

            <input type="password" name="password" placeholder="Password">
          <div id="button">
         <input  type="submit" name="submit" value="Submit">
        </div>
       </form>
```

Figur 2.3 visar koden för login formuläret.

```
<?php
//checks if a user is logged in.
//If it's true show logout button.
//If it's false show login button.
                if(isset($_SESSION['CurrentUser']))
                {
                        include "../../php/php/logoutButton.php";


                }
                else
                {
                        include "../../php/php/loginButton.php";
                }
```

Figur 2.4 visar koden för implementering av login och logout knapparna. Koden finns i varje recept sida.

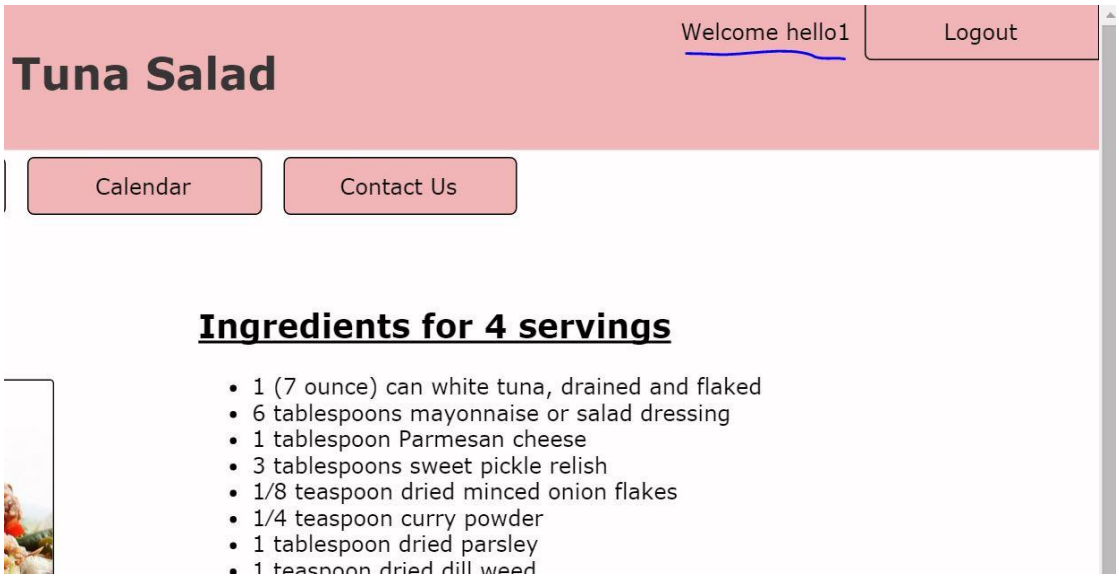
```php
//If the submit button is clicked validate given username and password from the login form
//with all usernames and passwords from MySQL database. Return either 1 or 0 where 1 is true and 0 is false.
if(isset($_POST['submit'])){

$username= mysql_escape_string($_POST['username']);
$password= mysql_escape_string($_POST['password']);

//Decrypts md5 password taken from the database.
$password= md5($password);
//default username = root with no password
$sql="SELECT * FROM `users` WHERE `username` = '$username' AND `password` = '$password'";
$result= mysql_query($sql, $connection);

if(mysql_num_rows($result) > 0){
   $_SESSION['CurrentUser'] = $username;
   include $_SESSION['currentPage'];
}
//If the validation failed restart login page with a given error.
else {

  include 'loginPage.php';
  echo'<div id="textecho"> Wrong username or password, try again!</div>';

}
```

Figur 2.5 visar funktionen för inloggningen.

```php
<?php
//End session when logout button is clicked and include(go to) previous page.
session_start();
$save=$_SESSION['currentPage'];
session_unset();
session_destroy();
unset($_SESSION);
include $save;
  ?>
```

Figur 2.6 visar funktionen för logout knappen.

* Figur 2.7, 2.8 och 2.9 visar vad som händer när en användare är inloggad.



Figur 2.7 visar logout knappen med Välkomsttext i en recept sida när användaren är inloggad.

- 1 tablespoon dried p
- 1 teaspoon dried dil
- 1 pinch garlic powd

Instructions

1. In a medium bowl, stir together the tuna, mayonnaise, Parmesan cheese, and onion fla
2. Season with curry powder, parsley, dill and garlic powder. Mix well and serve with crack

Comments

Comment:

comment

hello1:

Yummy!

Delete Edit

Figur 2.8 visar kommentarssektionen när användaren är inloggad.

```
<!--A simple comment form-->
<form action ="../../php/php/commentBox.php" method="POST">

   <table>

       <tr><td colspan="2">Comment: </td></tr>
       <tr><td colspan="2"><textarea name ="comment"></textarea></td></tr>
       <tr><td colspan="2"><input type="submit" name="submit" value="comment"/></td></tr>
    </table>
```

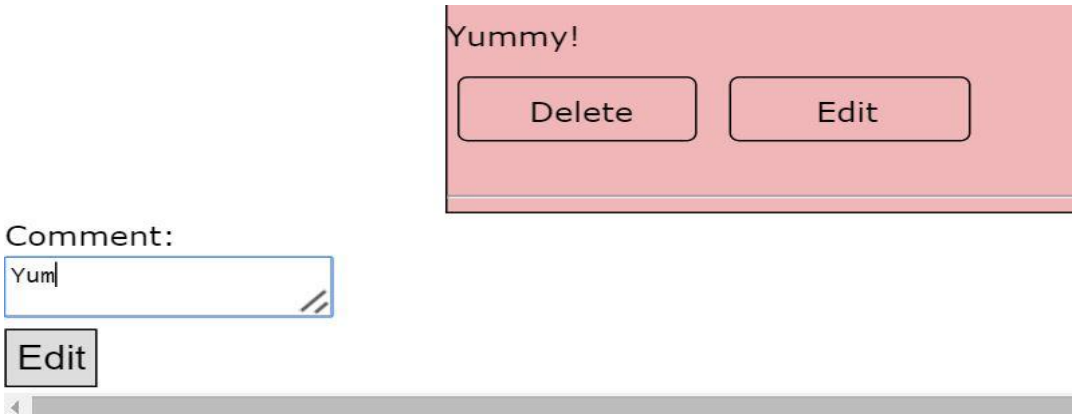
Figur 2.9 visar koden för kommentarformuläret.

Uppgift 3 Redigera kommentarerna:

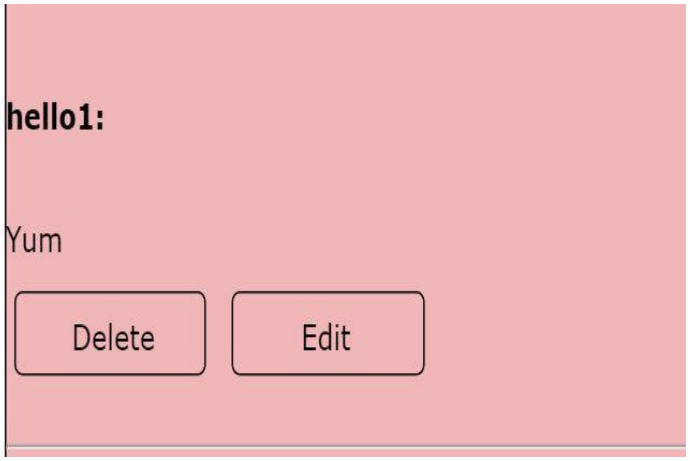
Den inloggade användaren har tillåtelse att redigera hans egna kommentarer.

```
<!--A simple edit form which contains the correct comment taken from the database to be edited by the user-->
<form action ="../../php/php/Edit.php?id=<?php echo$_GET['id'] ?>" method="POST">
      <table>
         <tr><td colspan="2">Comment: </td></tr>
         <tr><td colspan="2"><textarea name ="edit"><?php  echo$editComment ?></textarea></td></tr>
         <tr><td colspan="2"><input type="submit" name="submit" value="Edit"/></td></tr>
      </table>
   </form>
```

Figur 3.1 visar koden för redigeringsformuläret.



Figur 3.2 visar edit formuläret när användaren är inloggad i en recept sida.



Figur 3.3 visar att kommentaren har ändrats från figur 3.2.

```php
if(isset($_POST['submit']))
{
    //include code (connecting to database).
      include '../../php/php/databaseConnection.php';
   $updateid= $_GET['id'];
  $update = $_POST['edit'];

    $sql = "UPDATE comments SET comment = '$update' WHERE id= '$updateid'";
      mysql_query($sql, $connection);

      // Includes a saved page which is the previous page.
    include $_SESSION['currentPage'];
}

?>
```

Figur 3.4 visar en metod för att uppdatera databasen med den redigerade kommentaren.

```php
if(isset($_POST['submit']))
{
    //include code (connecting to database).
      include '../../php/php/databaseConnection.php';
   $updateid= $_GET['id'];
  $update = $_POST['edit'];

    $sql = "UPDATE comments SET comment = '$update' WHERE id= '$updateid'";
      mysql_query($sql, $connection);

      // Includes a saved page which is the previous page.
    include $_SESSION['currentPage'];
}

?>
```

Figur 3.5 visar funktionen för att redigera en kommentar. Resten av Edit funktionen finns i Figur 2.2.

Valfri uppgift 1, Registrera nya användare

Figur 4.1 visar registreringssidans UI design.

```php
require_once $_SERVER['DOCUMENT_ROOT'] . '/Config.php';
require_once URL . '/data/core/init.php';

use Util\Input;
use Util\Token;
?>

<div id="loginDesign">
<form action="/data/register.php" method="post">
    <div class="field">
        <label for="username">Username</label>
        <input type="text" name="username" id ="username" value="<?php echo escape(Input::get('username'));?>" autocomplete="off">
    </div>

    <div class="field">
        <label for="password">Choose a password</label>
        <input type="password" name="password" id="password">
    </div>

    <div class="field">
        <label for="password_again">Enter your password again</label>
        <input type="password" name="password_again" id="password_again">
    </div>

    <div class="field">
        <label for="name">Enter your name</label>
        <input type="text" name="name" value="<?php echo escape(Input::get('name'));?>" id="name">
    </div>
    <input type="hidden" name="token" value="<?php echo Token::generate(); ?>">
    <input type="submit" value="Register">
</form>
</div>
```

Figur 4.2 visar registreringsformulärets HTML kod.

```php
require_once $_SERVER['DOCUMENT_ROOT'] . '/Config.php';
require_once URL . '/data/core/init.php';

use Model\User;
use Util\Validate;
use Util\Token;
use Util\Input;
use Util\Hash;

$user = new User;

if(Input::exists()){
    if(Token::check(Input::get('token'))){
        $validate = new Validate();
        $validation = $validate->check($_POST, array(
            'username' => array(
                'required' => true,
                'min' => 2,
                'max' => 20,
                'unique' => 'users',
                'string' => true
            ),
            'password' => array(
                'required' => true,
                'min' => 6
            ),
            'password_again' => array(
                'matches' => 'password'
            ),
            'name' => array(
                'required' => true,
                'min' => 2,
                'max' => 50
            )
        ));
```

Figur 4.3 visar validering av registreringen. Valideringen kollar om användarnamnet och namnet är mellan 2 och 50 karaktärer, om lösenordet har minst 6 karaktärer och om lösenordet matchar det andra lösenordet angivna lösenordet.

```php
        if ($validation->passed()) {
            $user = new User();
            $salt = Hash::salt(32);
            try{
                $user->create(array(
                    'username'  => Input::get('username'),
                    'password'  => Hash::make(Input::get('password'), $salt),
                    'salt'      => $salt,
                    'name'      => Input::get('name'),
                    'joined'    => date('Y-m-d H:i:s'),
                    'groups'    => 1
                ));


                include_once $_SESSION['currentPage'];

            }catch (Exception $e){
                die($e->getMessage());
            }
        } else{
            include $DIR . '/data/View/registerPage.php';
            foreach($validation->errors() as $error){
                echo $error, '<br>';
            }
        }
    }
}
```

Figur 4.4: När valideringen har passerat kommer användarnamnet, lösenordet och namnet att registreras i databasen. Funktionen errors() skriver ut alla misslyckade valideringar.

```php
<?php
namespace Util;
use Integration\DB;

/*Essential class on any page  when:  user is registeration, changing password,  profile update
 */

class Validate{
    private $_passed = false,
            $_errors = array(),
            $_db = null;
    //Called when validation class is instantiated to checking if instance of database connection already exists.
    public function __construct(){
        $this->_db = DB ::getInstance();
    }

    /*Passing in the data we want to loop through and check with an array of predefined rules against their provided sources
     * and add to the errors as we go
     */
    public function check($source, $items = array()){
        foreach($items as $item => $rules){
            foreach($rules as $rule => $rule_value){

                $value = trim($source[$item]);
                $item = escape($item);

                if($rule === 'required' && empty($value)){
                    $this->addError("{$item} is required");
                } else if(!empty($value)){
                    switch($rule){
                        case 'min':
                            if(strlen($value) < $rule_value){
                                $this->addError("{$item} must be a minimum of {$rule_value} characters.");
                            }
                            break;
                        case 'max':
                            if(strlen($value) > $rule_value){
                                $this->addError("{$item} must be a maximum of {$rule_value} characters.");
                            }
                            break;
                        case 'matches':
                            if($value != $source[$rule_value]){
                                $this->addError("{$rule_value} must match {$item}");
                            }
```

Figur 4.5: Valideringsklassen.

```php
//The ability to create a user
public function create($fields = array()){
    if(!$this->_db->insert('users', $fields)){
        throw new Exception('There was a problem creating an account.');
    }
}
```

Figur 4.6: Funktionen create() som använder sig av insert() funktionen i DB klassen för att skapa ny användare. Funktionen finns i User klassen.

```php
//Inserting new data into database
public function insert($table, $fields = array()){
    $keys = array_keys($fields);
    $values = '';
    $x = 1;
    foreach($fields as $field){
        $values .= '?';
        if($x < count($fields)){
            $values .= ', ';
        }
        $x++;
    }
    $sql = "INSERT INTO {$table} (" . implode(', ', $keys) . ") VALUES ({$values})";


    if(!$this->query($sql, $fields)->error()){
        return true;
    }

   return false;
}
```

Figur 4.7: Insert() funktionen lägger in en array i databastabellen.

5 Kommentarer om kursen

Vi har spenderat minst 4 timmar varje dag åt uppgifterna vilket blir totalt 36 timmar inklusive föreläsningarna och handlerdingarna.

Det som skulle kunna förbättras i kursen är att dela upp handledningarna i två dagar. Det kan hända att man får problem eller vill ha hjälp vid flera tillfällen i veckan och det blir svårt att hantera det eftersom man bara har ett tillfälle at lösa problemen och få hjälp.