

PHP implementation

Introduktion

Denna text har för avsikt att beskriva arbetet av implementationen av en webbapplikation med hjälp av PHP. I webbapplikationen är det främst fokus på att hämta, lägga till och förändra data i en databas. Webbapplikationen skall alltså koppla upp sig mot en databas och därmed ska man kunna manipulera data därifrån. Med andra ord: uppgiften är skapa en webbplats som fungerar som en sammanhängande enhet med hjälp av databas och prepared statements (pdo).

1 Länk till en fungerande webbsida.

<https://wwwlab.webug.se/databaskonstruktion/a21liltr>

Inloggning sker med användarnamnet **'agent'** och lösenordet **'foo'**.

2 Formulär för att addera data

På hemsidan finns det flera sätt att både se och manipulera data, varav ett av dessa är att man kan registrera en alien med fritext. Se bilder nedan.

Register an ALIEN

Name: **7**

test

Home Planet: venus

2

ADD ALIEN

List of Aliens

Registered Aliens (fully registered aliens):

1010 Ove 20231017-0004 Ixion

3333 Bert 20231017-0002 Ixion

[Delete Alien Ove with ID: 1010](#)

[Delete Alien Bert with ID: 3333](#)

Figur 1: Fritext ruta med innehåll, innan klick på knapp.

Register an ALIEN

Name: 1

Home Planet: 2

List of Aliens

Registered Aliens (fully registered aliens):

1010	Ove 20231017-0004 Ixion
3333	Bert 20231017-0002 Ixion
Cgn87nVMAKDt2XVkedbRSzQmx test	20231026-0003 venus
1	2

Figur 2: Fritext ruta med innehåll, efter klick på knapp. Ny data synlig i listan "Registered Aliens"

Resultat syns i en ny sida där en text bekräftar att en ny alien har lagts till. Vid backning till föregående sida syns en uppdaterad lista med den nya alien.

Här används en form som skickar data till en annan fil som heter add_alien.php när en knapp klickas.

```
<h3>Register an ALIEN</h3>
<table>
  <form method="post" action="add_alien.php">
    <tr><td>Name:</td><td><input type="text" name="namn"></td></tr>
    <tr><td>Home Planet:</td><td><input type="text" name="hemplanet"
placeholder="Add a Home Planet to fully register"></td></tr>
    <tr><td><input class="execute" type="submit" value="ADD
ALIEN"></td></tr>
  </form>
</table>
```

Figur 3: Form med method POST och action som beskriver vilken fil som ska köras sen

Här finns input element med attributen 'name', 'name' sätts till samma namn som de kolumner som ska hämtas från databasen.

I varje php fil i projektet finns följande kod med:

```
<?php require "db_connection.php";
```

Figur 4: Databasanslutning

```

<?php
$host = "localhost";
$username = "a211ilitr_administratör";
$password = "bar";
$dbname = "a211ilitr";

try {
    $pdo = new PDO("mysql:host=$host; dbname=$dbname", $username,
$password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}

?>

```

Figur 5: Databas credentials

Detta för att filen "db_connection.php" är en väsentlig fil som är ansvarig för att upprätta en anslutning till databasen som har implementerats för denna uppgift.

```

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $alien_id = generateRandomString(25);
    $namn = $_POST['namn'];
    $hemplanet = $_POST['hemplanet'];
    $query = "";

    if(!empty($_POST['hemplanet'])) {
        $query = "INSERT INTO Registrerad Alien (alien_id, namn, hemplanet)
VALUES ('$alien_id', '$namn', '$hemplanet')";
    }
    else {
        $query = "INSERT INTO Oregistrerad_Alien (alien_id, namn) VALUES
('$alien_id', '$namn')";
    }

    $stmt = $pdo->prepare($query);
    $stmt->execute();

    echo "Alien added successfully!";
}

function generateRandomString($length) {
    $characters =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
    $string = '';
    for ($i = 0; $i < $length; $i++) {
        $string .= $characters[rand(0, strlen($characters) - 1)];
    }
    return $string;
}

```

Figur 6

I php scriptet kallas om metoden som körs är en POST. Den sätter sedan en variabel till en sträng med 25 slumpmässiga karaktärer. Detta blir en aliens nya ID. Sedan hämtas värdena som har satts in i de tidigare nämnda inputsen.

En SQL query defineras för att göra en insättning i databasen med den önskade datan. Om input elementet för hemplanet inte är tom, ska då insättningen göras i tabellen Registrerad_Alien, annars görs den i Oregistrerad_Alien. Ett meddelande ges om adderingen lyckades.

```
echo "<h3>List of Aliens</h3>";
echo "<h4>Registered Aliens (fully registered aliens): </h4>";
echo "<table>";
$query = "SELECT * FROM Registrerad_Alien";
$aliens = $pdo->query($query);
if($aliens != null)
{
    foreach ($aliens as $row)
    {
        echo "<tr>";
        echo "<td>" . $row['alien_id'] . "</td>";
        echo "<td>" . $row['namn'] . "</td>";
        echo "<td>" . $row['pnr'] . "</td>";
        echo "<td>" . $row['hemplanet'] . "</td>";
        echo "</tr>";
    }
}
echo "</tbody>";
echo "</table>";
```

Figur 7

Sist hämtas en lista med registrerade aliens och oregistrerade aliens. Här visas hur listan för Registrerad_Aliens hämtas. På samma sätt hämtas även en lista från tabellen Oregistrerad_Alien. Det visas inte eftersom det är nästan exakt likadant, bortsett från de hämtade kolumnerna.

3 Dropdownlista



The image shows a web form with a dropdown menu. The dropdown is titled "Races in the database" and has a downward arrow. It contains four options: "Ixionian", "Marsian", "MWDXACJAian", and "venusian". Below the dropdown is a text input field with the label "Name:".

Figur 8: Dropdownlista

Hemsidan innehåller även en dropdownlista med raser som existerar i databasen, exkluderat hemligstämplade samt "tomma" (NULL) raser.

```

<?php
    // Shows list of races in a dropdownlist,
    // Races are DISTINCT.
    try {
        $query = $pdo->prepare("SELECT DISTINCT ras_namn FROM
Offentliga_Raser_view");
        $query->execute();

        $stmt = $query->fetchAll(PDO::FETCH_ASSOC);

        if ($stmt) {
            echo "<select name='dropdown'>";
            echo "<option hidden disabled selected>Races in the
database</option>";
            foreach ($stmt as $row) {
                echo "<option>{$row['ras_namn']}</option>";
            }
            echo "</select><br>";
        } else {
            echo "<option hidden disabled selected>No races in the
database</option>";
        }

    } catch (PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
?>

```

Figur 9

Här hämtas rader från en view som heter Offentliga_Raser_view. Här hämtas enbart kolumnen ras_namn där alla rader måste vara distinkta, det vill säga att varje rad måste vara unik. Detta görs genom att förbereda en SQL query.

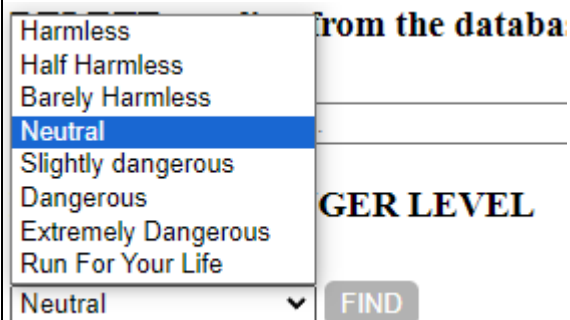
Alla rader sparas i en array som heter \$stmt.

Om \$stmt inte är tom, då visas en dropdownlist som har en 'hidden' och 'disabled' alternativ som ska visa texten "Races in the database". Detta alternativ går inte att välja eller se när dropdownlistan klickas på.

För varje rad som finns skapas en option element att lägga radens innehåll i. Om det inte finns några raser att hämta, visas istället texten "No races in the database", vilket igen, inte är ett valbart alternativ.

4 Sökning i databasen (+VG)

Innan:



from the databa

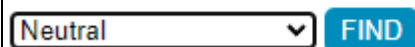
DANGER LEVEL

Neutral

FIND

Markerad:

Find Alien by DANGER LEVEL



Neutral

FIND

Efter klick på 'FIND':

ALIEN ID : DANGER LEVEL : RACE

1010	4	Ixionian
1111	4	
2222	4	MWDXACJAian
3333	4	Ixionian
4444	4	Marsian
7777	4	
8888	4	
9999	4	
Cgn87nVMAKDt2XVkedbRSzQmx	4	venusian
QR894uVbDo1HdMV58SbmKHaYi	4	venusian

Figur 10: Figur med markerade data och lista med alien id, farlighetsnivå, och ras

Här kan man även söka på data utifrån en dropdownlista. I detta fall kan man söka på en farlighetsgrad där en ny sida återger en lista med en aliens unika ID, dess farlighetsnivå samt dess ras.

```

<h3>Find Alien by DANGER LEVEL</h3>
<form method="post" action="search_by_danger.php">
  <select name="farlighet">
    <option value="1">Harmless</option>
    <option value="2">Half Harmless</option>
    <option value="3">Barely Harmless</option>
    <option value="4">Neutral</option>
    <option value="5">Slightly dangerous</option>
    <option value="6">Dangerous</option>
    <option value="7">Extremely Dangerous</option>
    <option value="8">Run For Your Life</option>s
  </select>
  <input class="execute" type="submit" value="FIND">
</form>

```

Figur 11: en option med en "value" attribut. Value:n skickas med efter att en submitknapp med "FIND" trycks.

Formen som håller SELECT-elementet skickar en POST request till search_by_danger.php filen när knappen av submittypen klickas. Därefter tar php scriptet emot datan från formen och konverterar värdet som skickades med för att säkerställa att det är en integer som hanteras.

```

$selectedOption = intval($_POST["farlighet"]); // Convert the value to integer,

$query = "SELECT * FROM Alien WHERE farlighet = :farlighet";

$stmt = $pdo->prepare($query);
$stmt->execute([
    ':farlighet' => $selectedOption
]);

if ($stmt->rowCount() > 0) {
    echo "<table>";
    echo "ALIEN ID : DANGER LEVEL : RACE <br><br>";

    while ($row = $stmt->fetch(mode: PDO::FETCH_ASSOC)) {
        echo "<tr>";
        echo "<td>" . $row['alien_id'] . "</td>";
        echo "<td>" . $row['farlighet'] . "</td>";
        echo "<td>" . $row['ras_namn'] . "</td>";
        echo "</tr>";
    }

    echo "</table>";
}

```

Figur 12: argumentet som skickas med i POST hanteras.

Det valda värdet i dropdownlistan nås genom `$_POST["farlighet"]` i php scriptet. Därefter skapas en SQL query för att hämta alla kolumner från tabellen Alien där farligheten är samma som :farlighet, vilket är en variabel som sätts till det mottagna värdet från formen.

If-satsen hanterar resultatet så att om det finns fler rader än 0 som returnerats, då går programmet vidare in i koden i if-satsen. While-loopen hämtar ut varje rad från resultatet och visas sedan i en HTML tabell som representerar varje tabelldata.

5 Förändring av innehåll i databasen med hjälp av UPDATE.

Reset Limit For Agent on Procedure

Johan AA 0 3

Kajsa BB 3 5

Kim BB 0 3

Lisa AA 0 3

Olof BB 0 3

Stina AA 0 3

Yngve AA 2 5

Agent:

Procedure name:

RESET LIMIT

Figur 13: Fritext ruta och reset knapp. Efter utförande har Kims användningar ändrats till 0.

I organisationen finns det ett tak på hur många gånger man får göra vissa ändringar i databasen därför finns det även på hemsidan en funktion för att nollställa en agents användningar av procedures. Genom att skriva in vilken agent och procedur det berör, nollställs datan för dessa. Användaren skickas till en ny sida och vid återvändandet till föregående sida syns resultatet på hemsidan direkt.

Först och främst finns en tabell/lista på agenter samt dess användningar på vissa procedurer med den första talet som representerar användningar och det andra talet som representerar en begränsning för proceduren. Denna lista hämtas på exakt samma sätt som i föregående kapitel, det vill säga, på samma sätt som hämtningen av `Regstrerad_Alien`.


```

<h3>Reset Limit For Agent on Procedure</h3>
<form method="post" action="reset_limit.php" >
  <table>
    <tr>
      <td><label for="agent">Agent:</label></td>
      <td><input type="text" id="agent" name="agent" required></td>
    </tr>

    <tr>
      <td><label for="kommando">Procedure name:</label></td>
      <td><input type="text" id="kommando" name="command"
required><br></td>
    </tr>
  </table>
  <input class="execute" type="submit" value="RESET LIMIT">
</form><br><br>

```

Figur 14

Här används återigen en form för att hämta datan genom POST metoden.

```

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['agent']) &&
isset($_POST['command']))
{
    $agent = $_POST['agent'];
    $kommando = $_POST['command'];

    $stmt = $pdo->prepare("CALL nollställ_begränsning(:agent, :command)");
    $stmt->bindParam(':agent', $agent, PDO::PARAM_STR);
    $stmt->bindParam(':command', $kommando, PDO::PARAM_STR);

    if ($stmt->execute())
    {
        echo "Limit reset successfully!";
    } else
    {
        echo "Something went wrong: " . $stmt->error;
    }
}
?>

```

Figur 15

På samma sätt utförs ett php script i en annan fil, denna gång i filen reset_limit.php. Scriptet börjar med att kontrollera parametrarna 'agent' och 'command'.

Efter att ha bekräftat att parametrarna är satta, hämtas deras respektive värden från \$_POST och tilldelas dem till variablerna \$agent och \$kommando. En SQL query definieras för att utföra den lagrade proceduren "nollställ_begränsning" där de hämtade värdena för 'agent' och 'command' sätts in som parametrar för proceduren.

Genom användning av metoden bindParam() binds de nämnda variablerna in i queryn.

Därefter utför scriptet det queryn, som kör proceduren i databasen. Ett villkorligt block implementeras för att hantera potentiella utfall: om utförandet lyckas, visar scriptet meddelandet "Limit reset successful!", om det uppstår ett fel printas i stället ett felmeddelande.

6 Exekvera en procedur

Unregistered Aliens (partially registered aliens) :

1010 Ove	20231019-215703
1111 Åsa	20231017-125740
<u>7777 Karo</u>	20231017-125740
9999 Gunilla	20231017-125740

DELETE an alien from the database

Alien with ID: 7777 has been deleted

Unregistered Aliens (partially registered aliens) :

1010 Ove	20231019-215703
<u>1111 Åsa</u>	20231017-125740
9999 Gunilla	20231017-125740

Figur 16: Radering av en alien på ID 7777 samt alla dess kopplingar

Genom att fylla i rutan körs en procedur som raderar en aliens alla kopplingar mot databasen. Om en alien har vapen, skepp, vare sig den är registrerad eller oregistrerad så kommer all information om den alien vars ID man fyller i att försvinna.

```
$id = $_GET['id'] ?? null;

if ($id) {
    try {
        $query = $pdo->prepare('DELETE FROM Registrerad_Alien WHERE
alien_id = :id');
        $query->execute([':id' => $id]);

        echo "Alien with ID: " . $id . " has been deleted";
    } catch (PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}
```

Här används återigen en HTML form för att skicka data till en annan fil som kör ett php script likt de andra implementationerna. Detta php script börjar med att hämta id parametern genom en `$_GET` denna gång. I php filen förbereds sedan en enkel DELETE query, därefter printas ett meddelande att den alien vars id man har fyllt i har raderats.

7 Visa innehåll från olika databastabeller på webbsidan.

Tabell över Procedure_Begränsningar:

Johan AA 0 3
Kajsa BB 3 5
Kim BB 0 3
Lisa AA 0 3
Olof BB 0 3
Stina AA 0 3
Yngve AA 2 5

Tabell över Aliens, varav de är kategoriserade efter registrerade/oregistrerade (dessa även olika subtabeller från Alien tabellen):

List of Aliens

Registered Aliens (fully registered aliens):

1010 Ove 20231017-0004 Ixion
3333 Bert 20231017-0002 Ixion
[Delete Alien Ove with ID: 1010](#)
[Delete Alien Bert with ID: 3333](#)

Unregistered Aliens (partially registered aliens) :

1010 Ove 20231019-215703
1111 Åsa 20231017-125740
7777 Karo 20231017-125740
9999 Gunilla 20231017-125740

Figur 17: Här syns två tabeller. Dessa uppdateras även vid förändring av data genom webbsidan.

En beskrivning av hur tabellerna hämtats har getts.

8 Generera länkar som tar bort rader från tabeller

Före:

List of Aliens

Registered Aliens (fully registered aliens):

1010 Ove 20231017-0004 Ixion
3333 Bert 20231017-0002 Ixion
bd9C7ZDcMiQEMb55TQv0EGe1x test 20231026-0003 venus
[Delete Alien Ove with ID: 1010](#)
[Delete Alien Bert with ID: 3333](#)
[Delete Alien test with ID: bd9C7ZDcMiQEMb55TQv0EGe1x](#)

Efter:

Register an ALIEN

Name:
Home Planet:

List of Aliens

Registered Aliens (fully registered aliens):

1010 Ove 20231017-0004 Ixion
3333 Bert 20231017-0002 Ixion
[Delete Alien Ove with ID: 1010](#)
[Delete Alien Bert with ID: 3333](#)

Figur 18: Här ser man hur test med långt ID försvinner i "efter" delen.

På hemsidan kan man göra en "snabb" radering av en alien. Här kan man klicka på en länk som är kopplat mot en registrerad alien i stället för att behöva skriva in hela det unika ID:t manuellt. Vid klick på länk kommer den registrerade alien och alla dess kopplingar att raderas från databasen. Resultatet syns omedelbart att den påverkade alien inte finns kvar i listan.

```

$query = $pdo->prepare('SELECT alien_id, namn FROM Registrerad_Alien');
$query->execute();
$aliens = $query->fetchAll(PDO::FETCH_ASSOC);

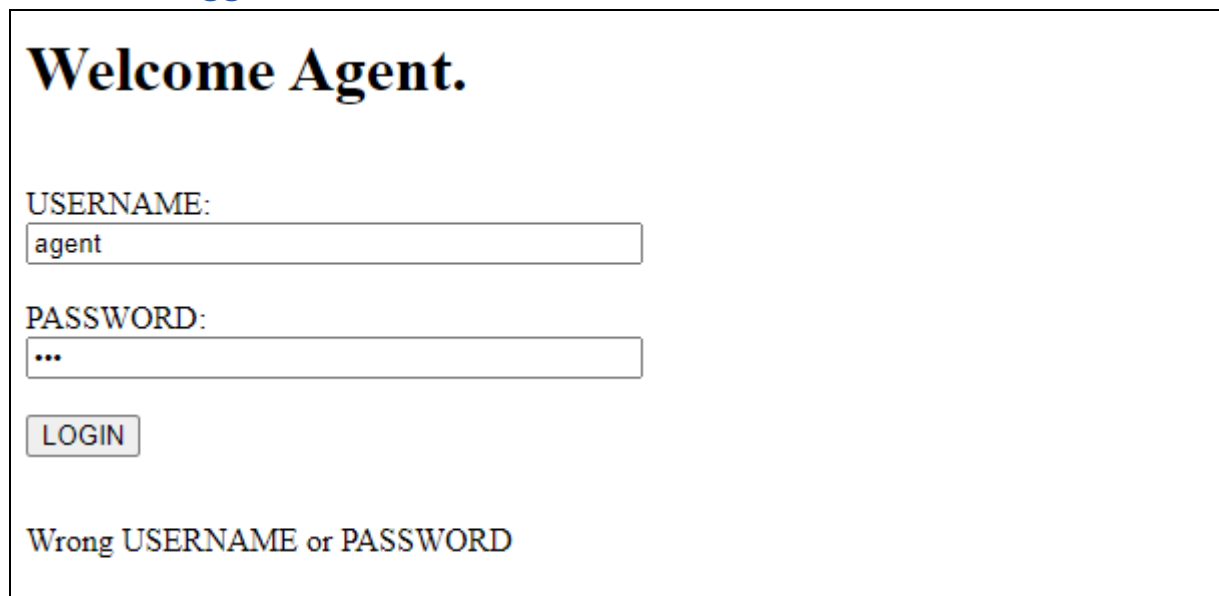
foreach ($aliens as $alien) {
    $id = $alien['alien_id'];
    $namn = $alien['namn'];

    //HYPERLINK to immediately delete a registered alien from the DATABASE.
    echo "<a href='delete_alien.php?id={$id}'>Delete Alien {$namn} with ID:
{$id}</a><br>";
}

```

Koden för hyperlänkarna implementeras i samma fil som visar länk elementen. En SQL query förbereds. Den gör en SELECT på alien_id och namn från Registrerad Alien. För varje rad skapas en <a href> element, och värdet på variabeln \$id sätts till det som hämtats. När länken klickas, körs php scriptet som finns i delete_alien.php där värdet av id:t är bifogat och ska används som parameter. Texten inuti beskriver vilken alien och id det är som kommer att raderas.

9 VG: Inloggsida



Welcome Agent.

USERNAME:

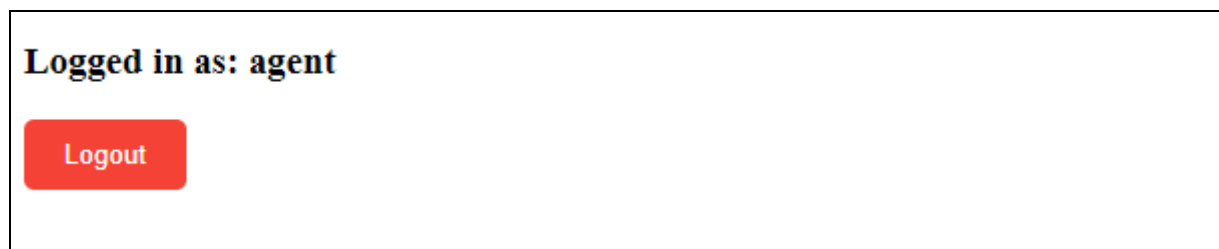
PASSWORD:

Wrong USERNAME or PASSWORD

Figur 19: Inloggningsida

Även en inloggningsida har skapats för webbplatsen.

I figur 19 ovan, har lösenordsrutan fyllts i med '123', vilket är fel, vilket då visar texten "Wrong USERNAME or PASSWORD" till användaren.



Logged in as: agent

Figur 20: Lyckad inloggning

Vid rätt inmatning av USERNAME och PASSWORD kombination kommer användaren till en ny sida där det står vem som användaren är inloggad som, samt en 'logout' knapp.

```
function loginUser($pdo, $username, $password)
{
    $query = "SELECT * FROM användare WHERE användarnamn = :username AND
lösenord = :password";
    $stmt = $pdo->prepare($query);
    $stmt->execute([
        ':username' => $username,
        ':password' => $password
    ]);

    return $stmt->rowCount() != 0;
}
```

Figur 21: Sätter värdet som har fyllts i input elementet som kolumnerna i databastabellen 'användare'.

Inloggningssidan är väldigt grundläggande och inkluderar en form som hämtar användarnamn och lösenord för autentisering.

```
try {
    if ($_SERVER['REQUEST_METHOD'] === 'POST' )
    {
        $username = $_POST["username"];
        $password = $_POST["password"];

        if($username && $password && loginUser($pdo,$username, $password))
        {
            $_SESSION['USER'] = $username;
            header('Location: landingsite.php');
        }
        else
        {
            echo "Wrong USERNAME or PASSWORD";
        }
    }
}
```

Figur 22: Kollar om någon rad med den specifika kombinationen existerar.

Värdena som fylls i input elementen jämförs med en tabell som ser till att rätt kombination av användarnamn och lösenord existerar i databasen.

Skulle det inte finnas en sådan kombination som har fyllts i input elementen, får användaren ett felmeddelande som säger att användarnamnet eller lösenordet är fel.

Om det är något fel på kopplingen mot databasen får användaren även här ett felmeddelande.