Applikation PHP

Erik Gustafsson (a23erigu)

Table of Contents

Intro

Denna rapport är skapad för inlämningen applikation PHP från kursen Databaskonstruktion G1F.
Inlämningen handlar om en hemsida som skapats åt en databas med hur denna hemsida är kopplad till och påverkar databasen.
Rapporten kommer främst fokusera på lösningar till problem som uppgiften ber en att lösa.
Endast php och html har använts till detta projekt och till php användes PDO för att koppla till databasen.
VG försöktes nå med detta program så de uppgifterna kommer tas upp i slutet.

Länk till webbsidan

https://wwwlab.webug.se/databaskonstruktion/a23erigu/Inlogg.php

Denna länk tar en till inloggning sidan där man loggar in och sedan tar sig till huvudsidan TomteVerkstad.php.

Användarna som finns är:
”dbkonstruktion” med lösenord ”Skata#23” (root användare).
” a23eriguByggarNisse” med lösenord ”ByggaBil” (kan vara så att de förlorat rättigheter).

Lägga till data i tabeller

Första delen som kommer gås över är hur information läggs till i databasen.
Resultatet av förändringen ska gå att se i sökningen av tomtenissar (namnet på den nya tomtenissen ska vara i sökfältet direkt efter skapandet så bara att klicka på sökknappen).

```
<form action = "TomteVerkstad.php" method = "POST">
        <p> Create tomtenisse </p>
        Name of tomtenisse
        <input type="text" name="Name" value=""> <br>
        ID of tomtenisse
        <input type="text" name="CId" value=""> <br>
        Amount of nuts earned
        <input type="number" name="Nuts" value=""> <br>
        Amount of raisin earned
        <input type="number" name="Raisin" value=""> <br>
        <input type="submit" name="submit" value="Submit">
</form>
```

Figur 1: html för infogandet av data.

Denna kod snutt är i html och finns för att kunna ta in data som behövs för att kunna skapa Tomtenissar. Alla fält måste ha något värde för att en tomtenisse ska kunna skapas och ID måste vara i formatet 000000-0000-0-000000000.
POST används för alla tillfällen där information ska tas med i programmet (med ett undantag) för att kunna uppnå lite bättre säkerhet.

```
If (isset($_POST["Name"]) && isset($_POST["CId"]) && isset($_POST["Nuts"]) &&
isset($_POST["Raisin"])) {

        $CreateTomtenissar = $pdo->prepare("insert into Tomtenisse(Namn,
        IdNr, Nötter, Russin) values(?, ?, ?, ?)");

        $CreateTomtenissar->bindParam(1, $_POST["Name"],
        PDO::PARAM_STR);

        $CreateTomtenissar->bindParam(2, $_POST["CId"], PDO::PARAM_STR);

        $CreateTomtenissar->bindParam(3, $_POST["Nuts"], PDO::PARAM_INT);

        $CreateTomtenissar->bindParam(4, $_POST["Raisin"],
        PDO::PARAM_INT);

        $CreateTomtenissar->execute();
```

Figure 2: säkerhets koll och infogande i databasen.

Här är php koden och den börjar med att kolla att alla POST måste vara med för att kunna köras.
Om all information är med så kör det sedan en prepare på en insert i databasen. Frågetecknen i inserten byts sen ut med hjälp av bindparamater till de informationen de ska innehålla. Till sista så används execute för att köra inserten.
Anledningen prepare används är för att stoppa SQL injektions från att kunna hända.

```
if (($CreateTomtenissar->rowCount()) > 0){
                echo "<br>";
                echo("Insert successful");
}
else{
                echo "<br>";
                echo("Insert failed");
                echo "<br>";
                echo $CreateTomtenissar->errorCode();
                echo "<br>";
                print_r($CreateTomtenissar->errorInfo());
}
```

Figur 3: check på infogandet av Tomtenissar.

Sista delen av skapandet är en check på om det lyckades skapa en Tomtenisse genom att kolla om inserten påverka någon rad i databasen. Om det misslyckades så kommer programmet att skriva ut ett error meddelande med hjälp av errorCode och errorInfo vilket skriver ut error koden samt info om erroret.

Dropdown (Listbox) genererad från data

Dropdown menyn använda i programmet för att vissa vad namnet på alla Tomtenissar är så man vet vad som går att söka på.

```
$Dropdownissar = $pdo->prepare("CALL getNissar");

$Dropdownissar->execute();

$FetchDropdownissar = $Dropdownissar->fetchAll(PDO::FETCH_ASSOC);

$Dropdownissar->closeCursor();



echo "<p> Name of all tomtenissar </p>";



echo "<select name = 'dropdown'>";

foreach($FetchDropdownissar as $row) {

                echo("<option>" . $row["Namn"] . "</option>");

}

echo "</select>";
```

Figur 4: koden för dropdown.

Denna del av koden behöver inte ta någon information från användaren utan ska alltid köras när hemsidan hämtas.

Koden börjar med att göra en prepare statment på stored proceduren ”CALL getNissar”, detta hämtar alla nissar som finns och ska kunna användas av alla användare (kommer gå mer in i det i Exekvera en procedur delen). Efter att ha hämtat all information så görs en foreach för att gå igenom varje rad och skriva namnet på användare i dropdown menyn. Dropbox menyn skapas genom "<select name = 'dropdown'>” vilket skapar skjälvaste menyn medans i loopen så säts ”<option>” runt varje rad vilket skapar raderna i dropboxen.

Sökning i databasen

Sökningen använda för att kunna hitta information om någon tomtenisse genom deras namn.

```
<?php
if (empty($_POST["Name"])) {
		$name = "Name";
}
else {
		$name = $_POST["Name"];
}
?>

<form action = "TomteVerkstad.php" method = "POST">
		<p> Search for Tomtenissar </p>
		<input type="text" name="Name" value= <?php echo $name ?>>
		<input type="submit" name="submit" value="Submit">
</form>
```

Figur 5: hämta namn för sökningen.

Den första delen av koden är php och kollar om de fått in något namn. Har de inte det så sätter programmet namnet till ”Name”.
De andra delen av koden är html och skapar rutan där användare kan skriva in information. De unika här är inputen Name vilket har sitt värde sätt till namnet som användaren senast skrev in vilket göt att de sparar vad användaren senast sökt på. Name fältet byts också ut till namnet på nya tomtenissar om det skapats en och till tomtenissens namn om det ändrats och programmet ska ta fram de nya/ändrade informationen (för någon anledning dock så blir det den gammal informationen som visas så användaren får trycka på sök knapen igen för att få korrekt information).

```
if(isset($_POST["Name"])){

		$Tomtenissar = $pdo->prepare("select * from Tomtenisse where Namn =
		?");

		$Tomtenissar->bindParam(1, $name, PDO::PARAM_STR);

		$Tomtenissar->execute();

}
```

Figur 6: prepare statement för att hämta tomtenissar.

Här så skapas prepare statmentet för att kunna få ut informationen ur databasen och bind parameten för att sätta in namnet.

```
if (($Tomtenissar->rowCount()) > 0){
		foreach($Tomtenissar as $row) {
			// Tomtenisse skrivs ut
			// Länk för att tabort
		}
}
else{
		echo "<br>";
		echo("No info found");
}
```
Figur 7: skriva ut de hittade nissarna.

Sista delen av utskriften av tomtenissar börjar med att kolla om de får ut någon information. Om det får någon information så körs en loop för att skriva ut de tomtenissarna. Hur tomtenissarna skriv ut tas upp i Två olika databastabeller.
De skrivs även ut åt varje nisse en länk med nissens namn vilket tar bort nissen om klickad på (denna länk pratas mer om under Generera länkar delen).

Förändring av innehållet

Det fins två sätt att ändra i databasen, en update och en delete. Denna del kommer bara prata om updaten medan deleten tas upp i Generera länkar. Updaten finns för att kunna göra tomtenissar till chefnissar.

```
<form action = "TomteVerkstad.php" method = "POST">
        <p> Make chefnisse </p>
        Name of tomtenisse
        <input type="text" name="Name" value=""> <br>
        ID of tomtenisse
        <input type="text" name="ChefId" value=""> <br>
        Shoe size <br>
        <input type="radio" id="none" name="Shoesize" value="none">
        <label for="none">None</label> <br>
        <input type="radio" id="mini" name="Shoesize" value="mini">
        <label for="mini">Mini</label> <br>
        <input type="radio" id="medium" name="Shoesize" value="medium">
        <label for="medium">Medium</label> <br>
        <input type="radio" id="maxi" name="Shoesize" value="maxi">
        <label for="maxi">Maxi</label> <br>
        <input type="radio" id="ultra" name="Shoesize" value="ultra">
        <label for="ultra">Ultra</label> <br>
        <input type="radio" id="mega" name="Shoesize" value="mega">
        <label for="mega">Mega</label> <br>
        <input type="submit" name="submit" value="Submit">
</form>
```

Figur 8: hämta data för uppdatering.

Denna form tar in både namn och ID av tomtenissen som ska uppdateras och sedan ett par radio knappar för att bestämma nivån av chefnissen.

```
if ($_POST["Shoesize"] == "none"){
                $EditChef = $pdo->prepare("update Tomtenisse set Skostorlek = NULL
                where Namn = ? and IdNr = ? ");
                $EditChef->bindParam(1, $_POST["Name"], PDO::PARAM_STR);
                $EditChef->bindParam(2, $_POST["ChefId"], PDO::PARAM_STR);
                }
else{
                $EditChef = $pdo->prepare("update Tomtenisse set Skostorlek = ? where
                Namn = ? and IdNr = ? ");
                $EditChef->bindParam(1, $_POST["Shoesize"], PDO::PARAM_STR);
                $EditChef->bindParam(2, $_POST["Name"], PDO::PARAM_STR);
                $EditChef->bindParam(3, $_POST["ChefId"], PDO::PARAM_STR);
}
$EditChef->execute();
```

Figur 9: göra uppdateringen i databasen.

Här så körs uppdateringen till databasen genom update. Anledningen det finns två är för att skostorlek kan vara null (alltså inte en chefnisse) och html hade svårt att sätta en POST till null så fick göra på detta sätt.

```
if (($EditChef->rowCount()) > 0){
		echo "<br>";
		echo("Update successful");
}
else{
		echo "<br>";
		echo("Update failed");
		echo "<br>";
		echo $EditChef->errorCode();
		echo "<br>";
		print_r($EditChef->errorInfo());
}
```

Figur 10: check på uppdateringen.

Sista delen av uppdateringen är en check för om det lyckat med error meddelande om de misslyckats.

Exekvera en procedur

Det finns tre procedurs som används i programmet men bara en av de kommer tas upp här (den andra två tas upp i VG Exekvera en procedur med parameter).
En av proceduren som exekveras i programmet ligger i dropdown menyn och används för att hämta alla tomtenissar.

```
$Dropdownissar = $pdo->prepare("CALL getNissar");

$Dropdownissar->execute();

$FetchDropdownissar = $Dropdownissar->fetchAll(PDO::FETCH_ASSOC);

$Dropdownissar->closeCursor();
```

Figur 11: Exekvering av en procedur.

De första delarna är som en vanligt sql satser med ett prepare statment på stored proceduren och sen en execute. De unika som måste göras för procedurers är fetchAll och closeCursor. fetchAll används för att hämta informationen ur sql satsen och läger det i en ny variabel. closeCursor används sen för att stänga sql satsen. Anledningen detta måste göras är då stored proceduren inte tillåter någon annan sql sats att köras för än all information är hämtad. Kan också vara bra att veta att den nya variabeln med informationen är en array och inte en sql sats så kommandon blir annorlunda.

Två olika databastabeller

Två tabeller vissas på webbsidan, de första är Tomtenissar (har prats om tidigare i texten) och leksaker. På webbsidan går det att söka på leksaker beroende på pris. Mycket av koden för att söka på leksaker kommer tas upp i VG Exekvera en procedur med parameter.

```
<form action = "TomteVerkstad.php" method = "POST">

        <p> Get toys by prise </p>

        Price of toy

        <input type="number" name="Prise" value=""> <br>

        <input type="submit" name="submit" value="Submit">

</form>
```

Figur 12: hämta information för leksaker.

Allt som krävs för att få tag på leksaker är ett pris så formen kunde bli ganska liten.

```
echo("<table>");
        echo("<tr>");
                echo("<th> Name </th>");
                echo("<th> ID </th>");
                echo("<th> Weight </th>");
                echo("<th> Prise </th>");
        echo("</tr>");
        // resten av koden för att skriva ut leksaker
echo("</table>");
```
Figur 13: skriv ut leksakers rubrik.

Efter att leksaker hämtats (tas upp i VG Exekvera en procedur med parameter) så börjar programmet med att skapa ett tabel som all data ska ligga i. Den första raden i tablen är till för rubrikerna av varje data (dessa är hård kodande då de alltid ska vara samma).

```
echo("<tr>");
                echo("<td>");
                                print_r($row2["Namn"]);
                echo("</td>");
                echo("<td>");
                                print_r($row2["IdNr"]);
                echo("</td>");
                echo("<td>");
                                print_r($row2["Vikt"]);
                echo("</td>");
                echo("<td>");
                                print_r($row2["Pris"]);
                echo("</td>");
echo("</tr>");
```

Figur 14: skriv ut leksaker.

Detta är koden för att skriva ut varje leksak. Det ligger i en loop som kör för varje leksak de hittar och skapar då en ny tabell i tablen med den nya leksaken.

```
echo("<tr>");
                echo("<td>");
                        print_r($row[0]);
                echo("</td>");
                echo("<td>");
                        print_r($row[1]);
                echo("</td>");
                echo("<td>");
                        print_r($row[2]);
                echo("</td>");
                echo("<td>");
                        print_r($row[3]);
                echo("</td>");
                echo("<td>");
                        print_r($row[4]);
                echo("</td>");
                echo("<td>");
                        // länk för att ta bort
                echo("</td>");
echo("</tr>");
```

Figur 15: skriv ut tomtenissar.

Här är koden för att skriva ut Tomtenissar. Den är nästan samma som att skriva ut leksaker då det också använder en table och skriver ut sina rubriker i första raden. Det som är unikt för denna är att den använder nummer för att se vilken data som är vilken (det var så databasen skickat) och den har en länk på slutet viket är till för att ta bort.

Generera länkar

Det görs en länk till denna webbsida och den finns när man söker på tomtenissar för att kunna ta bort tomtenissar.

```
echo "<a
href='https://wwwlab.webug.se/databaskonstruktion/a23erigu/TomteVerkstad.php?Name=" . $row["Namn"] . "&Id=" . $row["IdNr"] . "'> Delete " . $row["Namn"] . " </a>";
```

Figur 16: länken för att ta bort.

Denna länk skaps när man söker på en användare och används för att ta bort den användaren. Länken går till samma sida som de är på fast de skickar med användare som ska tas bort namn och ID som GET vilket är enda gången GETS används i programmet. Det fick bli GETS här då det hittades något annat sätt att skicka med information i en länk.

```
$DeleteTomtenissar = $pdo->prepare("delete from Tomtenisse where namn = ? and IdNr = ?");

$DeleteTomtenissar->bindParam(1, $_GET["Name"], PDO::PARAM_STR);

$DeleteTomtenissar->bindParam(2, $_GET["Id"], PDO::PARAM_STR);

$DeleteTomtenissar->execute();
```

Figur 17: sql sats för att ta bort.

I början av programmet så kollar det om de fåt två GETS med namn och ID för att kunna ta bort en tomtenisse. Om de har får korrekt GETS så kör det sen denna sql satsen för att ta bort en tomtenisse och skriver sen ut om det lykades.

VG Inloggning

Inloggningen till websidan har sin egen php fil för at göra inloggningen enklare.
I programmet så används session för att spara inloggningen och använda det på andra sidan. Det finns även en move knap som tar en till nästa sida då det var problem med att få login knappen att byta till korrekt sida.

```
<form action = "Inlogg.php" method = "POST">
        <p> Login </p>
        Name
        <input type="text" name="Username" value=""> <br>
        Password
        <input type="password" name="Password" value=""> <br>
        <br>
        <input type="submit" name="submit" value="Login">
</form>
```

Figur 18: hämta inloggningen

Detta är html koden för att ta inloggnings informationen. Det unika här är lösenords infogningen vilket är av typen ”password” så det inte går att se vad som skrivits in.

```
try {
        $pdo = new PDO("mysql:dbname=TomteVerkstad;host=localhost",
        $_POST["Username"], $_POST["Password"]);
        $_SESSION["Username"] = $_POST["Username"];
        $_SESSION["Password"] = $_POST["Password"];
        echo "Login successfull";
        echo "<br>";
}
        catch (PDOException $e) {
        exit("Login failed: " . $e->getMessage());
}
```

Figur 19: checka och spara inlog

Här är vart inlogget testas och sparas så det går att använda senare.

```
try{
        $pdo = new PDO("mysql:dbname=TomteVerkstad;host=localhost",
        $Username, $Password);
        echo "<p>Loged in as " . $_SESSION['Username'] .  "</p>";
}
catch (PDOException $e) {
        exit("Login failed: " . $e->getMessage());
}
```

Figur 20: login i huvudfilen

Detta är koden som körs i början av huvudfilen för att kolla om inloggningen är ok.

```
<form action = "Inlogg.php" method = "POST">
		<input type="hidden" name="Logout" value="True">
		<input type="submit" name="submit" value="Logout">
</form>


<?php

if(!empty($_POST["Logout"])){
		$_SESSION["Username"] = "";
		$_SESSION["Password"] = "";
		echo "Logout successfull";
		echo "<br>";
}
?>
```
Figur 21: utloggningen

Denna kod är för att loga ut ur programmet. De använder sig av en hidden infogning för att skicka med att programmet ska logga ut genom att byta vad som är i session.

VG Exekvera en procedur med parameter

Denna del inkluderar det två andra procedurerna som fanns kvar i programmet. Dessa procedurers används till att hitta leksakerna till de andra tabellen som vissas.

```
$ToyByPrise = $pdo->prepare("Call getLeksakerPåPris(?)");

$ToyByPrise->bindParam(1, $_POST["Prise"], PDO::PARAM_INT);

$ToyByPrise->execute();

$FetchToyByPrise = $ToyByPrise->fetchAll(PDO::FETCH_ASSOC);

$ToyByPrise->closeCursor();
```

Figur 22: hämta leksaker

När leksaker ska hämtas så gör det väldigt likt till första proceduren fast ”Call getLeksakerPåPris” behöver en bindparameter för att sätta värdet.

```
$ToyName = $pdo->prepare("Call getNamnPåLeksak(?)");

$ToyName->bindParam(1, $row["IdNr"], PDO::PARAM_STR);

$ToyName->execute();

$FetchToyName = $ToyName->fetchAll(PDO::FETCH_ASSOC);

$ToyName->closeCursor();
```

Figur 23: hämta leksak namn

Den sista proceduren som används är direkt efter föra då den inte ger namnet på leksaker så denna måste användas för att få ut deras namn. Det är efter denna procedur som det skrivs ut i leksaker.