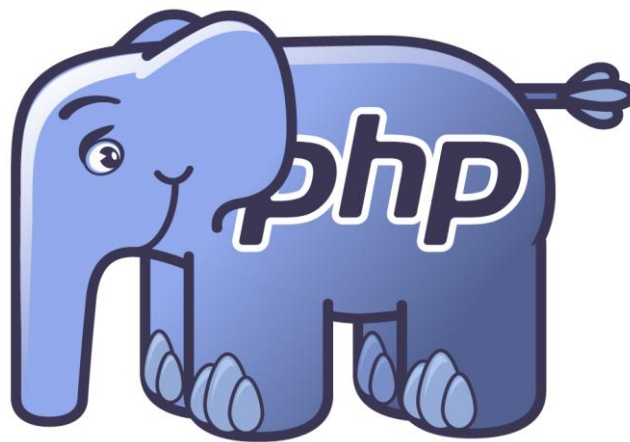


WEBBUTVECKLING - SERVERSIDAN

Distanskurs på grundnivå 7,5 ECTS

Alexander Krasse



Institutionen för data- och systemvetenskap (DSV)
Webbutveckling – Serversidan
Gesällprov
Grunder i PHP Vårterminen 2018

SAMMANFATTNING

Detta projekt är ett gesällprov i kursen Webbutveckling – Serversidan. För att kunna köra programmet måste en databas skapas. Detta kan exempelvis göras genom att importera den medföljande databasfilen med hjälp av phpMyAdmin.

Syftet med arbetet är att göra ett gesällprov. Kriterierna för gesällprovet är att det skall baseras på de tekniker som kursen bygger på och ska vara bra både till form och funktion.

Programmet har skrivits i Notepad++ v7.5.7 med språken HTML, CSS, JavaScript och PHP. För att kunna simulera en webbserver under utvecklingen har XAMPP använts, och datalagring (highscore) är gjord i MySQL med hjälp av phpMyAdmin. Förutom dessa har även JavaScript biblioteket jQuery använts.

Målet med projektet är att göra ett spel som liknar spelet "Mastermind" där målet är att gissa en av datorn framslumpad kombination. Spelet spelas i browsern där logik som rör själva spelet ligger på en server. Så att användaren ej har tillgång till den rätta lösningen förrän spelet är över. På servern skall det även finnas en databas som håller reda på de bästa resultaten.

INNEHÅLLSFÖRTECKNING

SAMMANFATTNING	2
INNEHÅLLSFÖRTECKNING	3
1. INTRODUKTION	4
1.1. Bakgrund	4
1.2. Syfte	4
1.3. Mål	4
1.4. Metod	4
2. GESÄLLPROV	5
2.1 Körning av programmet	5
2.2 PHP-Filer	5
2.2.1 Index.html	5
2.2.2 Home.php	6
2.2.3 rules.php	6
2.2.4 highscore.php	6
2.2.5 newgame.php	7
2.2.6 startGame.php	7
2.2.7 run.php	8
2.2.8 saveData.php	9
2.2.9 database.php	9
2.3 Text-filer	9
2.3.1 intro.txt	9
2.3.2 rules.txt	9
2.4 CSS-filer	9
2.4.1 MenuStyle	9
2.4.2 IndexStyle	10
2.4.3 GameStyle	10
2.5 JavaScript-filer	10
2.5.1 IndexScript.js	10
2.5.2 CreateNewGame.js	10
2.5.3 GameScript.js	10
2.6 Lagring av data	10
2.6.1 \$_SESSION	10
2.6.2 MySQL Databas	10
Bilaga 1. Filöversikt	12

1. INTRODUKTION

Detta projekt är ett gesällprov i kursen Webbutveckling – Serversidan. För att kunna köra programmet måste en databas skapas. Detta kan exempelvis göras genom att importera den medföljande databasfilen med hjälp av phpMyAdmin.

1.1 Bakgrund

IB909C Webbutveckling - Serversidan 7,5 hp är en kurs på Institutionen för data- och systemvetenskap (DSV) Stockholms Universitet. Kursen är en grundkurs som behandlar hur man skapar program för webbserver. Förutom övningsuppgifter ingår ett gesällprov där tekniker från kursen tillämpas.

1.2 Syfte

Syftet med arbetet är att göra ett gesällprov. Kriterierna för gesällprovet är:

- Ska baseras på de tekniker som kursen bygger på men får givetvis även innehålla tekniker från andra områden
- Ska vara bra både till form och funktion
- Ska inte vara en av de tidigare uppgifterna på kursen men får gärna vara en utökning av en eller flera av dessa.

1.3 Mål

Målet med projektet är att göra ett spel som liknar spelet "Mastermind" som går ut på att gissa en av datorn framslumpad kombination. Spelet spelas i browsern där logik som rör själva spelet ligger på en server så att användaren ej har tillgång till den rätta lösningen förrän spelet är över. På servern skall det även finnas en databas som håller reda på de bästa resultaten.

1.4 Metod

Programmet har skrivits i Notepad++ v7.5.7 med språken HTML, CSS, JavaScript och PHP. För att kunna simulera en webbserver under utvecklingen har XAMPP använts, och datalagring (highscore) är gjord i MySQL med hjälp av phpMyAdmin. Förutom dessa har även JavaScript biblioteket jQuery använts.

2. GESÄLLPROV

Målet med gesällprovet är att göra en hemsida där man kan spela ett "Mastermind"-liknande spel där spelets logik behandlas av servern så att användaren ej har tillgång till den rätta kombinationen.

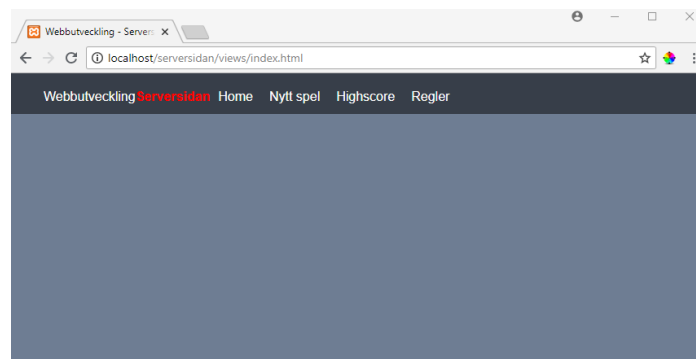
2.1 Körning av programmet

Hemsidan består av endast en sida med knappar som laddar in olika innehåll med hjälp av JavaScript och PHP. Innehållet ändras utan att uppdatera skärmen genom Ajax-anrop.

Länkarna berättar för en funktion i JavaScript som använder Ajax-anrop vilket PHP-script som ska anropas på servern. PHP-scripten skapar HTML-kod genom att läsa in från antingen .html och .txt-filer eller från databas. HTML-koden skickas som ett JSON-objekt tillbaka till JavaScript-funktionen på klientdatorn som anropade denna varefter denna HTML adderas till en <div>.

2.2 PHP-Filer

2.2.1 Index.html



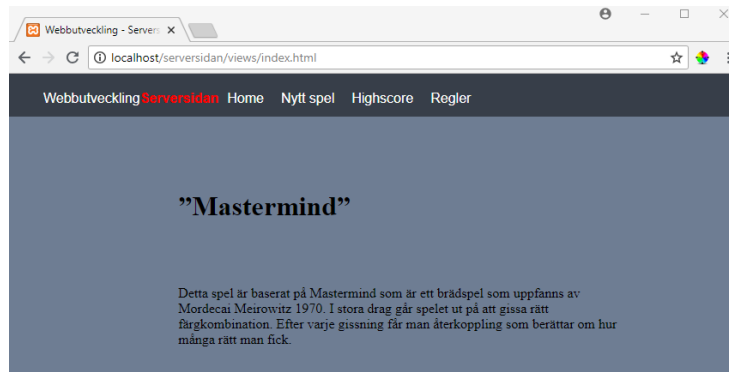
Figur 1. Index.html utan att något PHP-script har anropats.

Huvudvy som skapar en meny där man når alla delar av hemsidan. Menyvalen aktiverar ett JavaScript som kör olika PHP-script.

Här laddas även alla JavaScript- och CSS-filer in. Index.html innehåller en <div> vars innehåll beror på vad man valt i menyn och uppdateras via Ajax-anrop från de olika scripten. Som default-värde så anropas *home.php*.

2.2.2 Home.php

Läser in välkomsttext från *intro.txt* som läggs till i html-kod från *home.html*. Detta returneras till *Index.js* som uppdaterar innehållet i en `<div>` på huvudvyn.



Figur 2. *Index.html* där *home.php* har anropats

2.2.3 rules.php

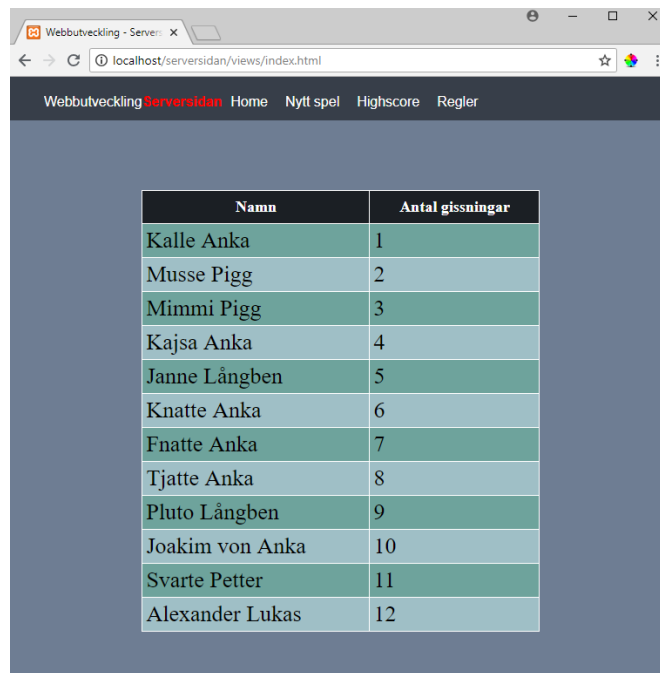
Läser in text om regler för spelet från *rules.txt* som läggs till i html-kod från *rules.html*. Detta returneras till *Index.js* som uppdaterar innehållet i en `div` på huvudvyn.



Figur 3. *Index.html* där *rules.php* har anropats

2.2.4 highscore.php

Skapar en tabell med html-kod som läses in från *highscorerows.html* och *highscoreheader.html*. Värdena i tabellen läses in från databas med hjälp av filerna *connectionvalues.txt* (innehåller data för att ansluta till databas/tabell) och *database.php* (som skapar en anslutning till databasen). Html-koden skickas sedan tillbaks till *IndexScript.js* som ett JSON-objekt som konverteras tillbaks till html-kod som sedan sätts som innehållet i en `<div>` på huvudvyn.

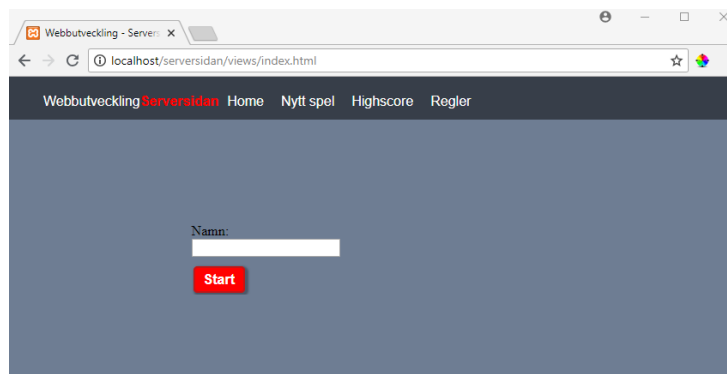


Namn	Antal gissningar
Kalle Anka	1
Musse Pigg	2
Mimmi Pigg	3
Kajsa Anka	4
Janne Långben	5
Knatte Anka	6
Fnatte Anka	7
Tjatte Anka	8
Pluto Långben	9
Joakim von Anka	10
Svarte Petter	11
Alexander Lukas	12

Figur 4. *Index.html* där *highscore.php* har anropats som laddar in värden från databas

2.2.5 newgame.php

Läser in html-kod från *newgame.html*. Html-koden skickas sedan tillbaks till *IndexScript.js* som ett JSON-objekt som konverteras tillbaks till html-kod som sedan sätts som innehållet i en `<div>` på huvudvyn. Html-koden innehåller inmatning för spelarnamn och en bekräftelseknapp som kör *createNewGame.js*.

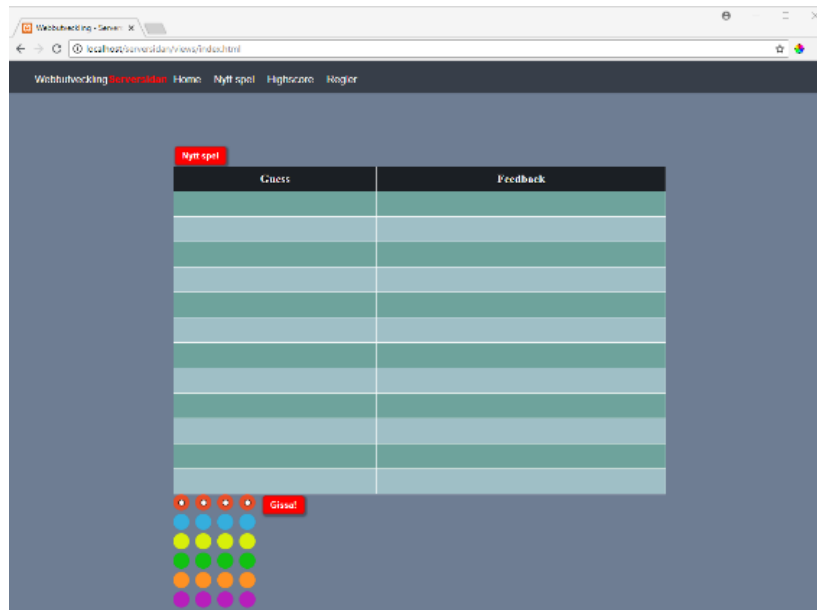


Namn:

Figur 5. *Index.html* där *newgame.php* har anropats

2.2.6 startGame.php

Skapar spelplanen genom att ladda in html-kod från *guesses.html*, *gameboard.html*, *gametable.html*. Varje cell i tabellen får ett eget ID för att senare kunna lägga till värden till rätt cell. Html-koden skickas sedan tillbaks till *IndexScript.js* som ett JSON-objekt som konverteras tillbaks till html-kod som sedan sätts som innehållet i en `<div>` på huvudvyn. I denna kod finns knapp för att köra *gamescript.js* eller *createNewGame.php*. *startgame.php* tar även fram 4 slumpade färger som motsvarar den rätta raden som användaren skall gissa sig till. Detta värde lagras i en sessions-variabel för att kunna användas vid varje gissning som användaren skickar till servern. Spelarnamnet och omgångsnumret lagras även de i sessions-variabler.



Figur 6. Tabell och radiobuttons som utgör spelplanen

2.2.7 run.php

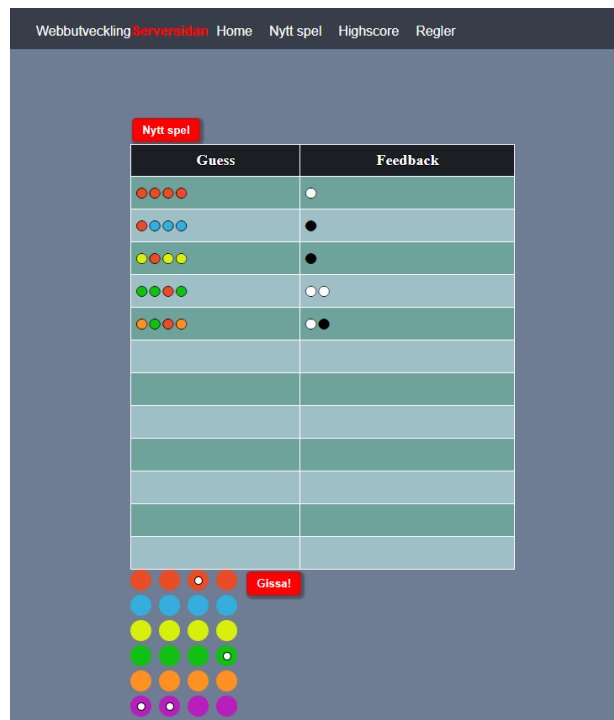
Hanterar själva spellogiken. Spelarens gissning skickas in som kontrolleras mot det rätta svaret och feedback ges på detta. Gissningen består av att användaren väljer mellan olika radiobuttons som ligger i en `<form>` som skickar radiobutton-värdet till *run.php* med metoden POST.

Kontrollerar hur många av varje färg som gissningen och svaret innehåller var för sig. Summan av det minsta värdet mellan svar och gissning för varje färg är antalet rätt man har i avseende på enbart färg.

Sedan kontrolleras antalet rätt färg där även positionen är rätt. Detta tal är antalet "vita" cirkelar som ges som feedback.

Antalet svarta cirkelar som innebär rätt färg men fel plats beräknas genom att ta det totala antalet rätta färger minus antalet vita cirkelar. De svarta och vita cirkelarna lagras som en sifferkombination som skickas till en funktion som skapar class-namn på ett `` beroende på vad för färg som skall sättas. Html-koden för `` och cirkel läses in från *indicator.html*.

Om användaren gissat rätt anropas *saveData.php* som lagrar resultatet i databasen.



Figur 7. Utseende under spelets gång

2.2.8 saveData.php

Skapar en anslutning till databasen genom att anropa *database.php*. Läser in tabell-namn från *connectionvalues.txt* och lägger till data till tabellen.

2.2.9 database.php

Skapar en anslutning till databasen. Servernamn, databasnamn, användare, lösenord, tabellnamn läses in från *connectionvalues.txt*.

2.3 Text-filer

2.3.1 intro.txt

Innehåller välkomsttext som läses in om man trycker på "Home"-länken

2.3.2 rules.txt

Innehåller text om hur man spelar, som läses in om man trycker på "Regler"-länken

2.3.3 connectionvalues.txt

Innehåller uppgifter för att ansluta till databasen. PHP-script som läser och skriver från databas använder denna fil för att slippa gå in och ändra värden för varje script som använder anslutningsinformation.

2.4 CSS-filer

2.4.1 MenuStyle

Innehåller CSS-kod som är kopplad till menyn.

2.4.2 IndexStyle

Innehåller CSS-kod som allmänt hör till webbsidan, d.v.s. inte har någon direkt koppling till menyn eller själva spelet som exempelvis `<table>`, `<body>` med mera.

2.4.3 GameStyle

CSS-kod för sådant som har direkt koppling till spelet. Som styling av radiobuttons som utgör valet av gissningar, och textfärger på feedback och gamla gissningar.

2.5 JavaScript-filer

2.5.1 IndexScript.js

Anropar `home.php` efter att DOMen lästs in. Innehåller Ajax-anrop för menyn, och gör om JSON-objektet som returneras till html-kod som appliceras till `<div>`.

2.5.2 CreateNewGame.js

Kontrollerar att spelaren angett ett namn. Anropar `startgame.php` som ritar ut spelplanen om namnkriteriet är uppfyllt.

2.5.3 GameScript.js

Skapar click-event för de två knapparna på spelplanen.

Innehåller funktion för att anropa `run.php` och skicka med vilka radiobuttons man valt. Svaret som returneras appliceras till `<div>`. Skriver ut rätt kombination om spelet är över och man inte gissat rätt. Den rätta kombinationen skickas ej från servern såvida inte spelet är över så detta skrivs ej ut om man lokalt skulle ändra i scriptet.

2.6 Lagring av data

Det rätta svaret är lagrat på servern och returneras endast när spelet är slut. För att lagra data har de två metoderna PHP `$_SESSION` och MySQL – databas använts.

2.6.1 \$_SESSION

`$_SESSION['name']` Lagrar användarnamnet

`$_SESSION['turn']` Lagrar hur många gissningar man gjort

`$_SESSION['answer']` Lagrar den rätta framslumpade färgkombinationen

Anledningen de två sista värdena lagras på servern är att användaren inte skall komma åt dessa.

2.6.2 MySQL Databas

Databas: kalleanka

Tabell: highscore

Tabellen består av 3 kolumner. Id som är nyckeln och autogenereras av phpMyAdmin, Name som är det namn användaren matat in, och Score.

Tabell 1. Hur tabellen Highscore är uppbyggd med några exemplvärden

Id int(10) primary	Name varchar(50)	Score int(5)
0	Kalle Anka	1
1	Musse Pigg	2
2	Mimmi Pigg	3
3	Kajsa Anka	4

Anrop görs från highscore.php som plockar ut de 12 bästa resultaten, saveData.php som skriver in nya värden till tabellen, och database.php som öppnar en koppling med databasen

Bilaga 1. Filöversikt

.html, SQL, .txt filer returnerar värden till .php som skickar ett json-objekt till scriptet som det anropades av. Som i sin tur applicerar html till huvudvyn (Index)

