DOKUMENTATION AV MY BOOK LIST, WEBBSERVERPROGRAMMERING

Av Richard Johansson

TE18IT – Webbserverprogrammering 1

Fredric Persson

Kattegattgymnasiet

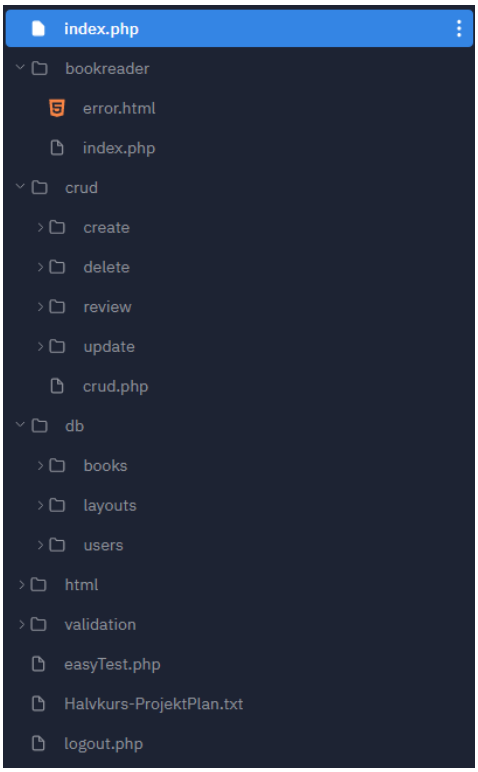
2020-12-16

Sammanfattning

Detta webbserver projekt är en typ av boklista där användare kan lägga till böcker i en allmän boklista som visas för alla medlemmar och icke medlemmar.

Innehållsförteckning

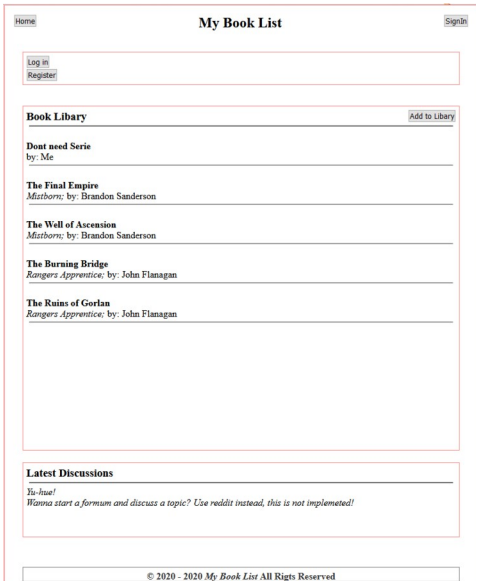
Mappstruktur & Huvudsida



Överst i strukturen ligger min index.php(blå-markerad), den utgör startsida och huvudsidan för detta projekt.

Index.php visar alla böcker som är inlagda i listan, tillsammans med kopplingar till att logga in och att registrera sig. Om man är inloggad ser sidan i princip lika dan ut bortsätt från att loggin och register-knapparna i första boxen är utbytt mot ett välkomstmeddelande kopplat till ens användarnamn (username).

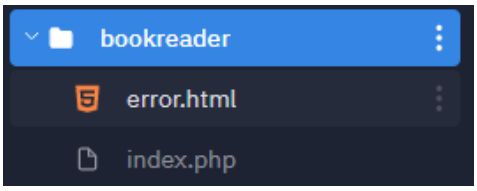
Bild 3 på denna sida visar koden för att söka igenom hela bok biblioteket för att hitta alla böcker i databasen. Den använder scandir för att hitta alla undermappar och lägger dem i en lista. Sedan tar jag listan, kollar om det finns en bok.json direkt i seriemappen, vilket innebär att boken bara har titel, annars söks seriemappen igenom och resultaten läggs därefter in i listan. Efter båda scandir-funktionerna använder jag array_diff för att ta bort punkterna som kommer med i arrayn.




```php
$bookIndex = [];
if ($series = scandir("db/books/")) { //Gets directory to all avalible books
    $series = array_diff($series, array('..', '.')); //delets the dots in start of scandirs, representing upper and current directory (i think)
    foreach ($series as $serie) {
        if (!file_exists("db/books/" . $serie . "/book.json")) {
            $books = scandir("db/books/" . $serie);
            $books = array_diff($books, array('..', '.'));
            foreach ($books as $book) {
                if (file_exists("db/books/" . $serie . "/" . $book . "/book.json")) {
                    $bookIndex[] = "db/books/" . $serie . "/" . $book . "/book.json";
                }
            }
        } else {
            $bookIndex[] = "db/books/" . $serie . "/book.json"; // puts directories of the book in an array
        }
    }
}
```

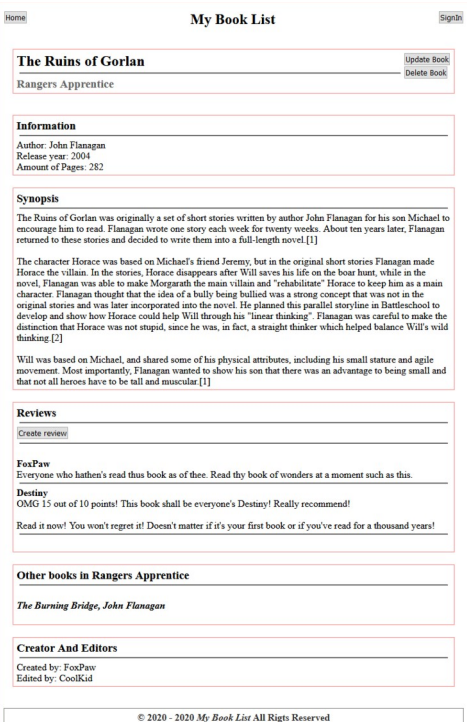
Richard Johansson
TE18IT – Webbserverprogrammering 1
HT - 2020

Bookreader



Bokläsaren och dess Index.php är det som läser alla böcker.

Index.php tar in en get-array med title och serie som ska läsas in och tar därefter och läser in det från databasen. All information hämtas från json-dokument.



I bokläsaren kan man uppdatera och ta bort böcker vilket visas bredvid titeln av boken. Knapparna syns även om man inte vore inloggad, men trycker man på någon av dem så länkas man direkt till inloggningssidan.

I böckerna kan man lägga in kommentarer, som även kan tas bort och det är endast skribenten som kan ta bort kommentarerna. Hos kommentarerna ligger även ett användar-id (uid) som är kopplat till alla konton. Användar-id tar och översätts med uidToUsername funktionen som från id tar fram användarnamn när kommentarerna renderas, detta är utifall att man någon gång skulle implementera användarnamns byte så kommer användarnamnen på kommentarerna att bytas ut automatiskt.

Efter kommentarerna finns det även kopplingar till andra böcker i serien, dem hittas genom att man skannar seriemappen med scandir(), därefter tas punkterna bort från listan. Där efter läggs böckernas fillänk in i bookindex som sedan öppnas av nästa del för att läsa ut titel och författarinformationen från filerna. På rad 98 kollas det om de andra böckerna är skilda från nuvarande titel, det är för att man i detta fallet inte vill länka till samma bok som redan visas. De andra böckerna i serien har även en länk som går att klicka på.

```
94      $bookIndex = [];
95      $books = scandir("../db/books/" . $serie);
96      $books = array_diff($books, array('..', '.'));
97      foreach ($books as $book) {
98          if (file_exists("../db/books/" . $serie . "/" . $book . "/book.json") && $book != $title) {
99              $bookIndex[] = "../db/books/" . $serie . "/" . $book . "/book.json";
100         }
101     }
102     $bookList = "";
103     foreach ($bookIndex as $book) {
104         $jsonBook = json_decode(file_get_contents($book));
105         $bookList = $bookList . file_get_contents("../html/linkedbook.html");
106         $bookList = str_replace("{{bookTitle}}", $jsonBook->title, $bookList);
107         $bookList = str_replace("{{bookAuthor}}", $jsonBook->author, $bookList);
108         //$tserie = strtolower($_GET['serie']);
109         $bookList = str_replace("{{bookPath}}", "?title=" . strtolower($jsonBook->title) . "&serie=" .
            $_GET['serie'], $bookList);
110     }
111     $page = str_replace("{{otherBooks}}", $bookList, $page);
112     $page = str_replace("{{tserie}}", $jsonBook->serie, $page);
113 } else {
114     $page = str_replace("{{tserie}}", "this Series", $page);
115     $page = str_replace("{{otherBooks}}", "", $page);
116 }
117
```

Sidan avslutas med skaparen av sidan samt dem som har uppdaterat sidan. Båda använder uidToUsername.

Om boken och serien som skickas med till bookreader inte hittas skickas man till error.html som bara säger att boken inte hittades.

uidToUsername & användarlagrings struktur

Denna funktion används bland annat i kommentarer, skapare och uppdaterar hos de olika böckerna.

```php
function uidToUsername($uid){
  $user = (array)json_decode(file_get_contents("../db/users/".$uid.".json"));
  $username = $user['username'];
  return $username;
}
```

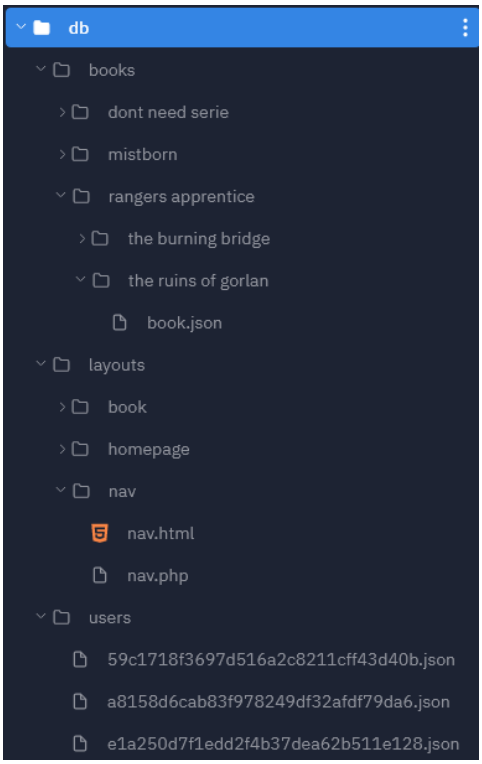
Den fungerar att man skickar in ett uid (användar-id) till funktionen som kollar i databasen efter användar-id:s fil för att sedan läsa ut användarnamnet från den filen.

Detta fungerar eftersom användarnas separata filer baseras på deras användar-id. Användar-id skapas originalet från att användarens email md5:as.

I användning kan den se ut såhär. Man hämtar uid från lista, skickar in dem i uidToUsername och får ut användarnamn som kan användas direkt i utskrift.

```php
foreach ($json->edit as $editor) { // if there are editors, t
    if ($editor == $json->edit[0]) {
        $editors = "Edited by: " . uidToUsername($editor);
    } else if ($editor == end($json->edit)) {
        $editors = $editors . " & " . uidToUsername($editor);
    } else {
        $editors = $editors . ", " . uidToUsername($editor);
    }
}
```

Databas (mappstruktur)



Mappstrukturen som motsvarar databasen innehåller 3 huvudmappar.

Böcker (books) inne håller böckernas filer. Mappstrukturen fortsätter med bokens serie-namn och därefter bokens titel. Om boken saknar serie skapas bara en titelmapp utan seriemapp. Detta gör det lätta att läsa fram böcker frånboknamnen.

Layouts innehåller templtates som har mer än endast html kopplat till sig. Exempelvis har nav.html en nav.php kopplat till sig som gör lite grejer.

Användare (users) är där alla användarnas data ligger. Deras filnamn baseras på deras mejl adress som md5:as.

Header (Location:) & exit ()

Efter varje utbyte av header har jag lagt till en “exit();”.

```php
//checks if book file is found, ie if book exists, else sends to error.html
if (!json_decode(file_get_contents("../db/books/" . $tserie . $title . "/book.json"))) {
    header("Location:error.html?title=" . $_GET['title'] . "&serie=" . $serie);
    exit();
}
```

Det beror på att på vissa ställen har det förekommit problem att man inte skickas iväg direkt vid en header, utan att koden fortsätter att arbeta. Detta var främst ett problem vid validation av användare, vilket innebar att även om en header borde skicka iväg en så gjorde den resterande av koden ändå.

Exit(); fungerar på ett liknande sätt som die(). Skillnaden är att die() stänger av kopplingen medans exit() behåller en typ av koppling.

Information hämtad från: https://stackoverflow.com/a/20932511/13380974

Exit() in action

```php
<?php
    header('HTTP/1.1 304 Not Modified');
    exit();
?>
```

Results in:

```
HTTP/1.1 304 Not Modified
Connection: Keep-Alive
Keep-Alive: timeout=5, max=100
```

Die() in action

```php
<?php
    header('HTTP/1.1 304 Not Modified');
    die();
?>
```

Results in:

```
HTTP/1.1 304 Not Modified
Connection: close
```

CRUD

Create, update, read, delete.

Create, update och delete har alla en egen htmlsida och php-sidan som består av dess funktionsnamn. Dessa 3 skickar en vidare till var sin procede.php som ligger i samma mapp som de olika funktionerna som gör vad originalfunktionen heter, det ända som procede.php gör att anropa rätt metod från crud.php för att sedan skicka en vidare till respektive bok.

Alla create, update och delete filer kollar om man är inloggad!

CreateBook.php skickar en till proced.php som anropar createBook funktionen från crud.php. Informationen som skickas in satineras och därefter skapas mappstrukturen med metodbilderna nedan. Sedan enkodas datan till jsonformat och sparas på filvägen som

```php
if ($serie != null) { //if dir doesn't exist and a series is set, create a new directory
    if (!file_exists("../../db/books/" . $serie . "/" . $title)) {
        $dir = "../../db/books/" . $serie . "/" . $title;
        mkdir($dir, 0777, true);
```

```php
else if ($serie == null) {//if dir doesn't exist and series isn't set, create a new directory
  if (!file_exists("../../db/books/" . $title)) {
      $dir = "../../db/books/" . $title;
      mkdir($dir, 0777, true);
```

skapades tillsammans med skapande av mappstrukturen.

UpdateBook.php skickar en till proced.php som anropar updateBook funktionen från crud.php. updateBook funktionen läser in den gamla datan, konverterar den till en array, och alla värden som inte är lika med de gamla värdena byts ut. Slutligen skrivs den gamla filen över med den nya filen.

DeleteBook.php skickar en till proced.php som anropar deleteBook funktionen från crud.php. DeleteBookfunktionen kollar om det finns flera inlägg i serien med en scandir metod. Om det finns andra titlar inlagda i serien skickas filvägen med titlemappen, annars skickas filmappen till hela serien till en metod som kallas för rrmdi. Det är en utökad rmdir funktion som tar bort objekt inklusive mappar. Den söker igenom mappen, kollar om innehaven i den är mapp eller objekt, vid mappar söks de igenom och man tar bort alla objekt och sedan mapparna. Anledningen till denna funktion är att den är baserad på rmdir, men rmdir kan inte ta bort

mappar om dem har ett innehåll, så därför sökas mapparna igenom som beskrivet ovan.

```php
function rrmdir($dir) { // removes direcroty-tree and everything inside it
    if (is_dir($dir)) {
        $objects = scandir($dir);
        foreach ($objects as $object) {
            if ($object != "." && $object != "..") {
                if (is_dir($dir . DIRECTORY_SEPARATOR . $object) && !is_link($dir . "/" . $object)) rrmdir($dir
                . DIRECTORY_SEPARATOR . $object);
                else unlink($dir . DIRECTORY_SEPARATOR . $object);
            }
        }
        rmdir($dir);
    }
}
```

Login-, Register- Validation & userChecker

Register.php laddar in htmlsidan och efteråt skickas man till registerVal.php. RegisterVal.php börjar med att checka så att inga värden är tomma, om inget är tomt görs en görs en md5 kryptering på mailen, md5kyrpteringen kommer att bli kontots unika id, och den jämföras mot övriga användares unika id för att se till att det unika id:t inte existerar. Efteråt sker en Password_hash på lösenordet med en kostnad på 14.

När lösernordet är haschat jämförs mailen och användarnamnet (med funktioner från userChecker som används av både login.php och register.php) mot alla andra användares namn och mejl, anledning varför mailen checkas efter att vi har checkad md5krypteringen av mailen är om man skulle implementera mailbyte har man lite mindre arbete i framtiden. Slutligen läggs md5:an av mejlen in i användarfilen och filen sparas som en ny jsonfil med namn av det unika id:t.

Login.php laddar in html-sidan, sedan skickas man till loginVal.php, där det checkas så att alla värden inte är tomma. Mailen md5:as vilket innebär att man har anändarfilen direkt, och behöver inte söka igenom hela systemet. Mailen i konton som ska loggas in på jämförs med mailen som är inskriven, sedan jämförs lösenorden med en password_verify.

Är allt godkänt startas en session och i sessionstaggen läggs att ‘loggedIn’ är sann, sedan läggs användarnamn och ens unika användar-id in. Slutligen skickas man till startsidan.

Logout

Logout.php anropas främst genom att mata in argumenten ?logout=logout i startsidan, detta är för att undkomma lite directories-problem i replit. Från startsidan skickas man till logout.php där en session förstörs och man skickas tillbaka till den sidan där man var innan man tryckte på logout knappen