U05-Car Rental

Sammanfattning:

För att allting ska funka så börjar det med att index.php-filen definerar databasanslutningen. Därefter så skapas en ny router och med en request(hämtar url:en och eventuellt formulärdata), routern kollar vad den url:en ska kopplas till för controller och metod i den kontrollen genom att den kollar vad som står i routes.json där det är definerat och kollar dessutom om det finns några eventuella parametrar. Den exekverar därefter den metoden i den controllern som den hittat. Controller-metoden i sin tur hämtar eller skickar eventuellt information som behövs via antingen formulärdatan ifrån föregående sida eller genom en funktion som den kallar ifrån en model-class. Model-classen i sin tur hämtar, uppdaterar eller tar bort information ifrån databasen och skickar eventuell information tillbaka till controller-funktionen. Därefter så renderar controller-funktionen den twig-filen som den blivit fördefinerad att rendera och skickar med eventuella properties som den behöver för att skapa sidan. Därefter så skickar den tillbaka den informationen till index.php some echo:ar ut den renderade twig-filen med dess properties på sidan.

Router:

Router-classen är den class som är till för att kolla vilken url(path) som webbläsaren är på och därefter dirigera vidare till den controller och metod som ska exekveras för att rendera rätt vy med rätt information.

Router-classens huvudfunktion:

route():
Funktion som använder sig av getPath funktionen i Request-classen för att hämta url:en som använder för nuvarande befinner sig på.
Den kollar därefter eventuella parametrar som t. ex. På denna url:
http://car.test/editCustomer/5702130161/Stefan%20Backenfeldt/Faltorp%2027/75223%20Uppsala/0767894256

Där /editCustomer är själva sidan i sig. Medans /5702130161/Stefan%20Backenfeldt är parametrarna.
Den hämtar därefter all information ifrån routes.json filen, decodar den och jämför sedan vilken controller-class och metod(funktion) i den controllern den ska använda sig av baserat på just den pathen den fått med sig.

Model:

Model-classerna är till för att hämta, ta bort eller uppdatera information i databasen. Alla funktioner i Model-classerna exekveras av controller-funktioner som kallas via routern. Alla Models är uppsorterade efter vilken del av projektet de har med att göra. T. ex. CustomerModel.php innehåller de funktioner som har med kunderna att göra när det kommer till databasen och historyModel.php innehåller de funktioner som har med history i databasen och vyn.

Följande typer av funktioner är några exempel:

customers():
Funktionen är till för att hämta all kunder från databasen för att visa dem i kundvyn.

addCar():
Funktion som är till för att sätta in alla värden på en ny bil som användaren har lagt till i databasen.

editCustomer():
Funtion som uppdaterar databasen med de nya uppgifterna som ändrats på en kund.

setDays():
Funktion som uppdaterar history table i databasen med antalet dagar som en bil hyrts ut baserat på tiden mellan checkOutTime och checkInTime som redan finns i databasen sen innan.

Controller:

Controller-classerna innehåller de funktionerna som exekveras av routern. Beroende på vilken path(url) användaren är på så letar router upp rätt controller-class och metod. Metoden som exekveras kör den eller de model-funktioner som behövs för att hämta, ta bort eller uppdatera informationen som ska ändras i databasen, skickar den med information till en model-funktion så har den fått den ifrån antingen formulärdata på sida eller via url:en. Efter att den exekverat den/de modeller som behövs så renderar den själva twig-sidan och skickar med de properties som twig-sidan behöver.

I och med att controller-funktionerna exekveras enbart av routern baserat på url:en så finns det då exakt lika många controller-funktioner som det finns sidor. Dvs. En funktion för att rendera en sida.

Följande typer av funktioner är några exempel:

customerAdded():

Funktion som exekveras av routern när url:en är car.test/customerAdded. Hämtar informationen ifrån addCustomer formuläret och exekverar sen addCustomer-funktionen i customerModel.php och skickar med informationen om den nya kunden till den. Renderar sedan vyn “customerAdded.twig” med den nya kundens namn och personnummer för att kunna använda det på bekräftelse-sidan sen.

viewHistory():

Funktion som exekveras av routern när url:en är car.test/history. Hämtar informationen som behövs för sidan genom att exekvera viewHistory-funktionen i historyModel.php och renderar sedan vyn “history.twig” med den informationen.

checkOut():

Funktion som exekveras av routern när url:en är car.test/chekOut. Hämtar först information om alla kunder genom att exekvera customers-funktionen via customerModel.php och sedan alla tillgängliga bilar genom att exekvera

checkOutList-funktionen via checkModel.php och renderar sedan vyn “checkOut.twig” med listor för kunderna och de tillgängliga bilarna.

View:

View är alla html sidor som faktiskt visas för användaren, dvs. en twig fil för varje sida. Enligt användarens perspektiv så är det ju som vilken vanlig html som helst men i det här fallet är den skriven med twig. Twig är enkelt beskrivet som html men med enklare funktioner som man kan använda som loopar och if statements för att kunna ändra sidan baserat på den informationen som den renderas med.

View renderas av controller-funktionerna och echo:as i sin tur sen ut via index.php.

Databasen:

Databasen är en MYSQL-databas. Den är uppbyggd av följande tabeller:

Customers:

Beskrivning: Tabell som innehåller alla kunder.

Tabellens nycklar:

Personnummer(ssNr) som primärnyckel som är en bigint för att kunna hantera det stora numret.

Namn(name) som är en varchar på max 256 tecken.

Adress(adress) som är en varchar på max 256 tecken.

Post adress(postalAdress) som är en varchar på max 256 tecken.

Telefonnummer(phonenumber) som är en varchar på max 10 tecken.

OBS!

Förinlagda värden finns för denna tabell för att inte allt ska vara tomt ifrån börja för användaren.

Lade dessutom in en kund med enbart personnummer “0” och inga andra värden för att kunna ha det som default value på personnummer på bilar som

inte är uthyrda och för att kunna ändra till det i historik-tabellen för att kunna ha kvar statistik men ändå ta bort kunden ifrån tabellen helt.

Cars:

Beskrivning: Tabell som innehåller alla bilar

Tabellens nycklar:

Regestreringsnummer(regNr) som primärnyckel som är en varchar på max 6 tecken.

Year(year) som är en int. Värdet är bilens tillverkningsår.

Price(price) som är en float för att kunna hantera om man sätter priset till t. ex. 299,99kr. Värdet är bilens kostnad per dygn.

Make(make) som främmande nyckel ifrån tabellen makes som är en varchar på max 256 tecken. Värdet är bilens märke.

Color(color) som främmande nyckel ifrån tabellen colors som är en varchar på max 256 tecken. Värdet innehåller bilens färg.

Personnummer(ssNr) som främmande nyckel ifrån tabellen customers som är en bigint. Värdet visar vilken kund som för närvarande hyr bilen. Om värdet är 0(default) så innebär det att bilen är ledig.

Checkout Time(checkOutTime) som är en datetime. Värdet innehåller när bilen senast checkats ut, finns där för att enkelt kunna hämtas för att kunna displayas på bilvyn.

OBS!

Förinlagda värden finns för denna tabell för att inte allt ska vara tomt ifrån börja för användaren.

Lade dessutom in en bil med enbart regNummer "Removed" för att kunna ändra till det i history tabellen när en bil tas bort för att kunna ha kvar statistiken men att bilen i sig tas bort från tabellen.

History:

Beskrivning: Tabell som innehåller all historik angående tidigare ut- och incheckningar.

Tabellens nycklar:

Regestreringsnummer(regNr) som främmande nyckel ifrån tabellen cars. Värdet innehåller information om vilken bil som hyrts ut.

Personnummer(ssNr) som främmande nyckel ifrån tabellen customers. Värdet innehåller information om vilken kund som har hyrt bilen.

Checkout Time(checkOutTime) som är en datetime. Värdet innehåller när bilen checkades ut.

Checkin Time(checkInTime) som är en datetime. Värdet innehåller när bilen checkades in.

Days(days) som är en float. Värdet innehåller hur många påbörjade dygn som bilen var uthyrt.

Cost(cost) som är en float för att kunna hantera på priset per dygn har en eller två decimaler. Värdet innehåller hur mycket totalkostnaden för uthyrningsperioden blev baserat på antalet dagar.

Makes:

Beskrivning: Tabell som innehåller alla tillåtna bilmärken för bilar.

Tabellens nycklar:

Make(make) som primärnyckel som är en varchar på max 256 tecken.

OBS!

Förinlagda värden har lagts till på denna tabell och kan inte ändras av användare själv.

Colors:

Beskrivning: Tabell som innehåller alla tillåtna färger för bilar.

Tabellens nycklar:

Color(color) som primärnyckel som är en varchar på max 256 tecken.

OBS!

Förinlagda värden har lagts till på denna tabell och kan inte ändras av användare själv.