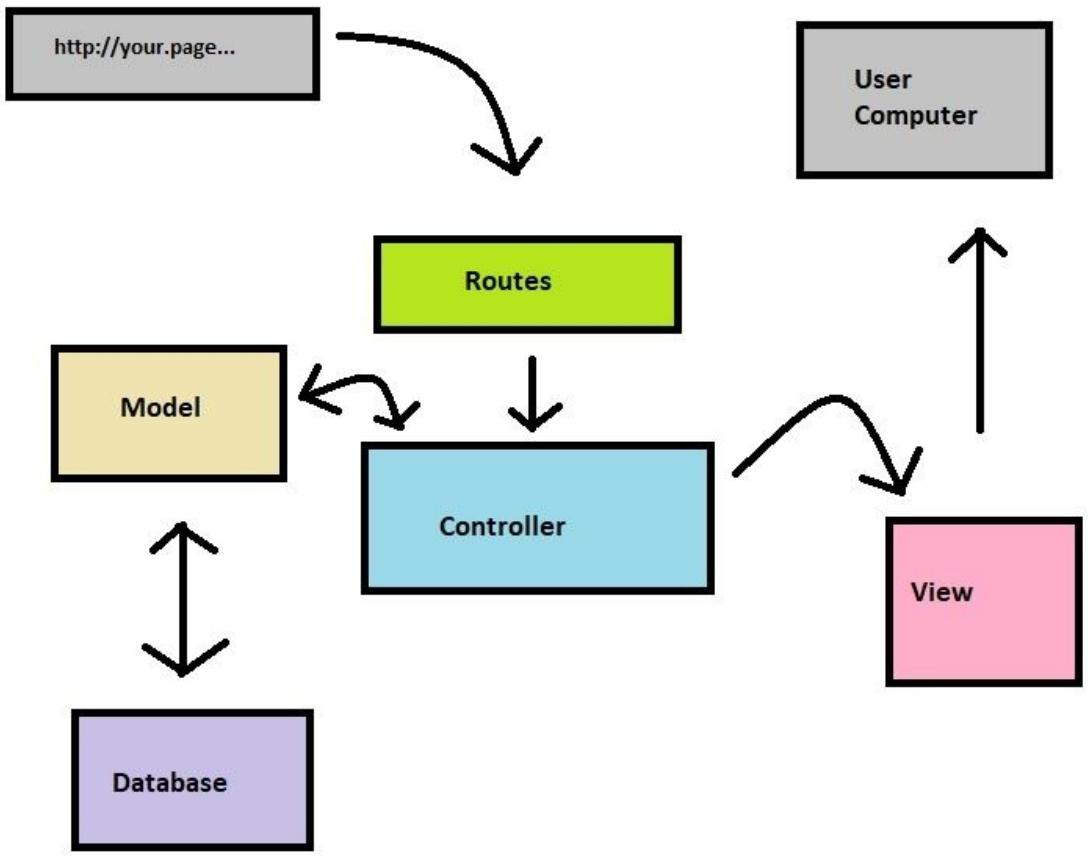
Rapport u05

MVC Modellen

Model - Strukturerar datan och förbereder den baserat på Controllerns instruktioner.
View - Visar datan i en användarvänlig vy baserat på användarens handlingar.
Controller - Tar in användarens instruktioner, skickar dessa till modellen för uppdatering av data och skickar sedan vidare instruktioner till View för att uppdatera denna. Kontaktar databasen.
Router - Dirigenten i strukturen, som hämtar eller skickar data till rätt enhet beroende på vad användaren väljer.



Ett exempel

Du går på krogen en kväll och ska beställa din favoritdrink, i detta fall en Manhattan.
Du går fram till baren, får kontakt med en bartender och säger “En Manhattan, tack!”
I detta fall är det du som är **user** och din begäran är en **user request**.
Bartendern nickar för att han/hon har förstått vad du begär, det är inte en god drink utan för honom/henne är det en serie steg som skall utföras.
Ta ett glas, addera whiskey, addera vermouth, addera is, rör om, addera ett körsbär och slutligen ta betalt. I det här exemplet är bartenderns hjärna **controllern.** När du anropar metoden “Manhattan” på ett språk som bartendern förstår börjar arbetet. Bartendern kan bara använda de verktyg som finns i baren, denna samling verktyg kallar vi här för **modellen** och innehåller dessa: Bartenderns händer, shakers/mixer, olika sorters sprit, drink mix, glas och garnering.
Slutligen, är drycken klar och du kan se och använda den. Detta är det som sker i **view**.
View är byggd från de verktyg som finns i modellen, arrangeras och fördelad via **controllern**.

Vill vi ha en ny drink? Skäller vi på det tomma glaset (**view)** händer inte så mycket, vi måste prata med bartendern (**controllern**) igen.
Likaså, vill man kanske inte att bartendern (**controllern**) heller alla ingredienserna direkt i munnen på oss (**user)** utan att han/hon blandar drinken åt oss i baren först. Dvs vi vill inte ha all logik i **view**, vi vill ha den i baren (**modellen**).
Om vi beställer en öl istället, behöver bartendern inte göra alls lika mycket. Men vi behöver såklart göra en beställning (**user request**) till bartendern. Ölen kommer ju inte magiskt dyka upp..

Nog om krogen. Tillbaka till webben.

- Användaren gör en begäran med en route, tex /rentalcar
- **Controllern** tar emot begäran och ger den en order som är relevanta för vår route. Det kan vara till vår **view** för att visa något för användaren eller till **modellen** för att utföra någon logik eller hämta något från en databas. Vi kan säga att vi vill hämta lite logik.
- **Modellen** kör logiken, hämtar från databasen och svarar **controllern**.
- **Controllern** skickar vidare denna logik och innehåll från databasen till vår **view** för användaren att se.

Min databas är byggd med tre tabeller.

- **Customers**, som har ett ID(person nr), Namn, Address, Post Adress och Telefon nr.
- **Cars**, som har ett ID (reg nr), Tillverkare, Färg, Årsmodell, Pris(dygnshyra) samt vem som hyrt bilen och när den lämnades för uthyrning.
- **History**, som visar reg nr, person nr, tid för utlämning, tid för inlämning, antal hyrda dagar samt kostnad för de dagarna.

Vid biluthyrning ska applikationen kolla om en bil är ledig genom en if-sats. Där den kollar en bil är uthyrd via tid/datum funktion. Är bilen tillgänglig ska man kunna boka den. Och om en bil är uthyrd ska man inte kunna ta bort den från databasen. Likaså med kunder, om en kund hyr en bil ska man inte kunna ta bort kunden ur registret. Historiken ska visas för varje moment som sker, uthyrning samt tillbakalämning av bil.