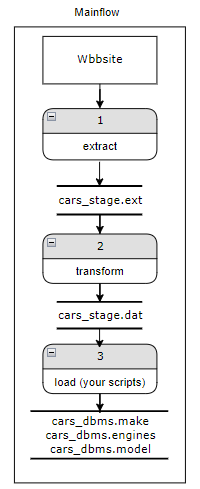
I denne opgave vil vi screenscrape en kilde på nettet og bruge de data vi finder til at lave en serie databaser, der i trin vil behandle de oplysninger vi har fundet. De enkelte trin i opgaven kan springes over, hvis teknikken volder dig kvaler er, er der i filer og beskrivelser der kan bringe dig videre.

Læringsmålene er:

1. Programmering: PHP til screenscraping.
   1. Screescraping teknologier i PHP
   2. No OOP
   3. Funktionel programmering – no sideeffects.
2. Etablering af staging databaser
   1. The BI way of doing it.
   2. Datatype og type konvertering,Completeness.
   3. Historik og batches.
   4. Fysisk og Logisk databaselag, hvem arbejder med hvad?
3. Etablering ad RDMS
   1. Normalformer og relationer
   2. DDL,DQL,DML

Målet er at lande vores screenscraping data i en RDBMS

**Opgaven kan i sin helhed løses individuelt, i gruppe, eller som en øvelse i pair-programming.**

Opgaven er beskrevet i afsnittet: [Etablering ad RDBMS](#_Opgave:_Etablering_ad)

Der hører to git repositorier til dette oplæg

<https://github.com/kimo1ucl/Webscraping>:

<https://github.com/kimo1ucl/SQL-Datatypes-Normalization>:

Contents

[Programmering: PHP til screenscraping. 2](#_Toc101136938)

[Etablering af staging databaser 2](#_Toc101136939)

[**Opgave**: Etablering ad RDBMS 3](#_Toc101136940)

[Instruks: PHP screenscraping 4](#_Toc101136941)

[Instruks: Etablering af stage og template database 5](#_Toc101136942)

[Stage database 6](#_Toc101136943)

[Template database 7](#_Toc101136944)

[Instruks: Anvendelse af template database 8](#_Toc101136945)

[Development Integration: Seed og Edit 8](#_Toc101136946)

[Development Integration: Generate database objects 10](#_Toc101136947)

[Konklusion om replika et BI udviklingsflow 13](#_Toc101136948)

[Afprøvning: Rådata ETL 13](#_Toc101136949)

# Programmering: PHP til screenscraping.

Sceenscraping er noget hvor PHP altid har skinnet! Specifikt til den disciplin har der længe eksisteret frameworks i PHP, som man i en Windows virksomhed måtte kigge langt efter. Vi ser på en interssant artikel der beskæftiger sig med at afprøve der forskellige teknologier.

<https://github.com/kimo1ucl/Webscraping>: Kode du kan downloade og prøve selv.

Men husk, at hvis der er forudsætninger som frameworks, så skal de også installeres for at download koden fungerer. Enkelte kræver installation af filer på dit operativ system, men mange danner alene filer i det projekt du arbejder med.

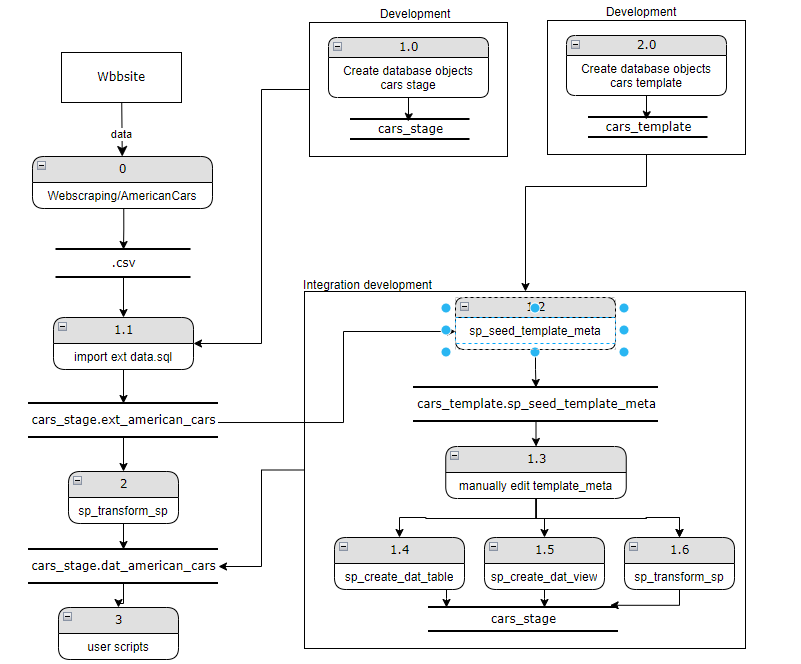
Når du installerer et framework, installerer du oftest kun filer i den mappe hvorfra du eksekverer. Der dannes en mappe: ’vendor’. I ’vendor’ mappen ser du mapper og kildekode, som det blev hentet/opdateret sidste gang kommandoen ’composer require <framework>’ blev kørt fra roden af webprojektet. Kræver disse framework installation af filer på operativsystemet, skal man være meget grundig med at registrere afhængigheden. Oplysninger af denne art er design dokumentatiom, og skal bruges hver gang en ny server i et givent miljø skal konfigureres.

# Etablering af staging databaser

Vi skal have importeret vores udtræk. I første omgang handler det om at få alle kollonner og alle rækker med. Vi importerer til en tabel vi vil kalde noget med ’ext’, så ved vi at dette er vores ’Extraction’ database. Alle kolonner er en overdimensioneret string type, fordi vi kompromiløst vil have data fra fil og ind i databasen. Derfra kan vi langt lettere lokalisere fejl og forekomster.

I det næste trin skal vi type konvertere, fordi i første omgang importerede vi bare det hele som strings.

* Kolonne for kolonne skal vi bestemme
  + passende datatype.
  + Om der er behov for datatranform

Database objekterne vi arbejder med i det videre forløb er views og stored procedures, i en replika af arbejdsgangen i et BI udviklingsflow hvor vi arbejder med rådata. Fra en ’cars\_template’ database genererer vi de nødvendige database objekter, der skal realisere en extract, transform, load proces så data lander i vores dat (for data) tabel.

I dfd diagrammet er det færdige load præsentret til venstre, til højre er er udviklingsindsats.

I repositoriet til denne del finder du komplet source kode, som du blot skal eksekvere mod din mysql server.

Læser du scriptsene ser du masser af SQL som du grundlæggende lærer fra w3school sitet mfl.

# **Opgave**: Etablering ad RDBMS

Vi skal have fundet sammenhænge i vores data. Klart at det handler om amerikanske biler i forskellige mærker, men givet at vi vil have data repræsenteret i en relationel database på 3. normal form, dvs. At vi forventer en tabel som feks. ’Make’ hvor der i rækkerne står Ford, Chevrolet, Cadillac. En anden tabel der helt sikkert også springer i øjnene er Model. Men hvordan med Make, Model og motortype? Ford har vel en serie motorer, men hvordan så med denne motor og de forskellige modeller? Er farveskemaerne Model eller Make specifikke?

**Du skal lave script(s) der kopierer data fra cars\_stage.dat\_ til tabeller, du også selv designer og danner, i cars\_rdbms databasen. Det kan være en god ide at dele scriptsene op i en ordnet sekvens, så du trinvist opbygger din samlede query.**

Målet er at lande vores screenscraping data i en RDBMS med jeres forståelse af disse sammenhænge.

* Der er ikke en sand løsning til opgaven, det handler også om de muligheder du ser i anvendelsen af data og den måde de repræsenteret på. Hvilken kardianlitet ser du gælder?.
* I opgaven med at etablere rdbms skal du alene tage udgangspunkt i de data der er tilgængelige.
* Selvfølgelig må du ikke opdatere rådata. Men hvis du vælger en løsning med pseudo nøgler (autoid) må de(n) nødvendige kolonner selvfølgelig tilføjes.
* Opgaven sigter ikke mod en tilgang, hvor du er fremsynet, og tænker i overordnet modellering, sådan at vi kunne have mange webscrabes, der med dine SQL scripts mellem stage og rdbms database, kunne gemme data fra alle webscrabes. Dette er en rådata extract-trensform-load. Når data skal herfra og til BI brugerne vil man måske lave datamarts, eller noget powerbi til at lave en tilsvarende extract-transform-load abstraktion.

# Instruks: PHP screenscraping

Denne instruks hjælper dig med at ekskvere php screenscrabing af et website tilhørende end amerikansk brugtvognsforhandler der udstiller brugte biler til salg.

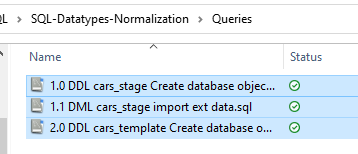
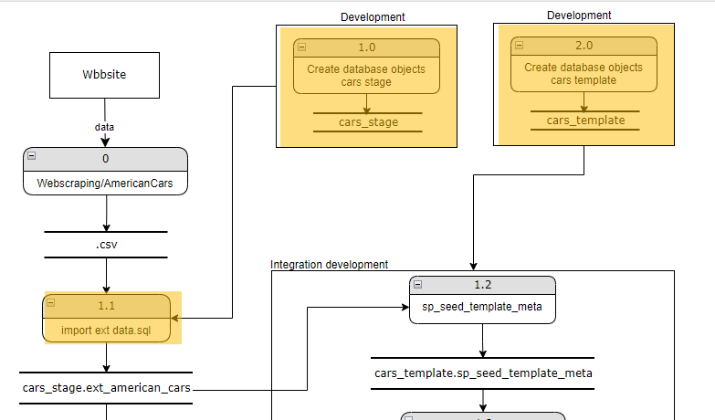
**Download kode fra:** [**https://github.com/kimo1ucl/Webscraping**](https://github.com/kimo1ucl/Webscraping)

|  |  |
| --- | --- |
| Åbn kode filen CarHelper.php og se at alle vores typer og instantieringer fejler. Vi har ikke det fornødne framework |  |
| Hvis du ikke har composer installeret skal dette gøre først | <https://getcomposer.org/download/> |
| Du kan køre PS scriptet cmposer.ps1 – alt det gør er at ekskvere disse to kommandoer i roden af projektet AMERICANCARS | composer require fabpot/goutte  composer require masterminds/html5  Alternativt skyd dem af I en dos command, igen i roden af projektet AMERICANCARS |
| Output ser du terminal vinduet, læs det igennem, der må ikke være errors, men warnings er som regel ok. Se også efter en konklussion på hver framework overførsel |  |
| ’composer require’ kommandoen danner vendor mappen, og composer.json og .lock |  |
| Kør ’CarRunner.php  En midlertidig fil dannes (’.AmericanCars.csv’)  Outputtet fra screenscrabet vises i debug console |  |
| Når scriptet slutter omdøbes filen med et timestamp |  |
| En kontrol kunne være at checke antal linjer i output fil. Der skal være en linje pr. annonceret bil + header. |  |

# Instruks: Etablering af stage og template database

Nu skal vi arbejde med databaser. Alle scripts er lavet klar til eksekvering., alt hvad du skal gøre er at eksekvere dem. Find dem i mappen **Queries**

Kildekoden kan du hente her: <https://github.com/kimo1ucl/SQL-Datatypes-Normalization>:



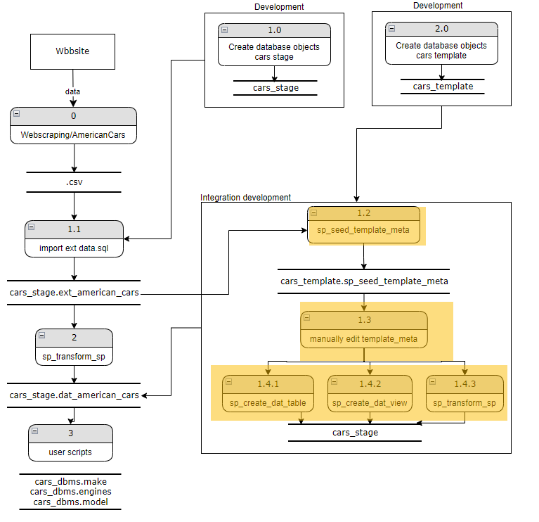
## Stage database

|  |  |
| --- | --- |
| Kør koden i filen | **1.0 DDL cars\_stage Create database objects.sql** |
| Check at der er oprettet em database  **cars\_stage**  og tabellen  **ext\_american\_cars** |  |
| Placer filen du vil importere i data mappen til din mysql installation. Bruger du xamp kan dette være et hint:  **C:\xampp\mysql\data**  Dette er en ”**beskidt**” måde at gøre det på, man kunne have brugt symbolske links feks. |  |
| Ideelt set ville alle dataudtræk ligge i en velordnet filstruktur, organiseret på en eller anden måde så man intuitivt ville finde filer for hver integration. Nogle integrationer indeholder måske mere end en fil feks.  Det man i denne øvelse skal vide er, at når der i sql filen står  LOAD DATA INFILE 'C:\20220411\_141816\_AmericanCars.csv'  Så indlæses fra:  C:\xampp\mysql\data\20220411\_141816\_AmericanCars.csv' | |
| Eksekver **’1.1 DML cars\_stage import ext data.sql’** tilpas din query så den matcher din input .csv fil. |  |
| Lav en select på tabellen:  SELECT \* FROM cars\_stage.ext\_american\_cars;   * Check rækkeantal * Check 1 datalinje I fil er den samme som første data linje I tabel * Check alle kolonner er mappet. |  |

## Template database

|  |  |
| --- | --- |
| Kør koden i filen | **2.0 DDL cars\_template Create database objects.sql** |
| Check at der er oprettet en database: **cars\_template**  og tabellen: **template\_meta**  og stored procedures som:  **sp\_create\_dat\_table**  **sp\_seed\_template\_meta**  Optionelt (simpel sp til intro)  **sp\_sample** |  |

# Instruks: Anvendelse af template database

I øvelsen her skal vi bestemme os for datatyper til vores screen scrabing data. Hvor lange skal tekstrengene være? Hvordan er numeriske værdier repræsenteret?

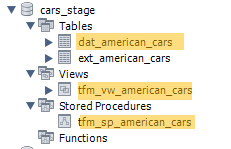
Hvis vi dimensionerer vores datatyper så de eksakt passer den længste streng dr måtte være repræsenteret i en given kolonne, risikerer vi at vi i et af de følgende udtræk står med data der ikke passer til vores dimensionering fordi den var for snæver. Det er ofte en skønssag hvor meget ekstra vi vil tillade. I nogle tilfælde vil det fremgå at data kan have intet eller lidt varians, mens andre kolonner måske rummer data med meget stor varians i længde.

Vi vil ”opdatere” vores beslutninger i tabellen template\_meta, derefter kan vi generere dataobjekter, der skal oprettes i stage databasen, og som skal anvendes hver gang data vi henter en ny datafil og data skal flyttes igennem ETL laget.

## Development Integration: Seed og Edit

|  |  |
| --- | --- |
| **1.2 sp\_seed\_template\_meta**  I template databasen kør kommandoen **sp\_seed\_template\_meta**`:  *CALL `cars\_template`.`sp\_seed\_template\_meta`('cars\_stage', 'ext\_american\_cars');*  Til højre er vist hvordan du med MySql Workbench kan højreklikke den stored procedure kan få genereret kommandoen. |  |
|  | |
| Nu skal du kunne se rækker i tabellen  **template\_meta**  I nogle kolonner er der mere varians end i andre.  Nogle kolonner ses indhold som indikerer at værdierne kommer fra en værdiliste. Lav feks. en select distinct på kolonnen data-fueltype:  *SELECT distinct `data-fueltype` from cars\_stage.ext\_american\_cars;* |  |
| Alternativt kan du bruge csv filen og **Excel** filtre til at identificere distincte forekomster så længe der er et overskueligt antal rækker i filen.  Hvis der er flere rækker end excel kan håndtere, kan **PowerBI** bruges  Men er der mange millioner rækker er **SQL** den eneste option. |  |
| **1.3 manually edit template\_meta**  Vi skal til at editere rækkerne i tabellen template\_meta.  At åbne en datatabel og editere felterne direkte kaldes data entry. Umiddelbart kunne jeg ikke finde featuren i MySql Workbench |  |
|  |  |
| Brug denne select når du du skal editere indholdet i tabellen templat\_meta  *SELECT \* FROM cars\_template.template\_meta*  *order by ordinal\_position;*  Data I ext tabellen løbes igennem, og vi indhenter oplysninger som min og max length, om der er tomme data i nogle kollonne/rækker. |  |
| Vi skal nu tilrette designet af vores dat tabel, ved at editere data i tabellen, direkte i Workbench Result grid | Kolonnen **data-vin** er temmelig sikkert en nøgle, altid 17 karakterer, datatypen ændres fra varchar til char (fast længde).  **data-year** og **data-price** er helt sikkert numeriske værdier, i dette tilfælde kan vi bruge en int som datatype. En int har ingen dimensionering, så felt i kolonnen dimension skal sættes til NULL.  **dimension** kolonnen ændres til en værdi der kan ligge over max length, som vi ser data repræsenteret i (i denne omgang), for at gøre interfacet mere robust og adresserende den usikkerhed der ligger i at basere design på et smalt udsnit af test data. |
| Her er mine bud på en forhåbentlig robust dimensionering.  Tryk på Apply når du er færdig. |  |
| I en dialog kan du se dine updates.  Tryk Apply  Updates i filen  2.1 DML UPdate template\_meta.sql |  |
|  | |

## Development Integration: Generate database objects

Her skal vi generere database objekter i cars\_stage databasen.

Vi ender op med en stored procedure, der kan kaldes med automatik eksekvering. Oracle og MSSQL server har selvstændige schedulere dedikeret denne use case.

Med MySql server vil man kunne lave en kommandolinje kommando afviklet i MySql CLI (command line interface) og lade operativ systemts scheduler klare opgaven.

Står MySql installationen på en Unix/Linux platform, ville crontab være et oplagt implementeringsrum for scheduleringen.

|  |  |
| --- | --- |
| **1.4.1 sp\_create\_dat\_table**  Kommandoen danner dat tabellen | *USE cars\_template;*  *DROP TABLE IF EXISTS `cars\_stage`.`dat\_american\_cars`;*  *CALL `cars\_template`.`sp\_create\_dat\_table`('cars\_stage', 'american\_cars');* |
| check at tabellen er designet som du forventer |  |
| **1.4.2 sp\_create\_tfm\_view**  Kommandoen danner et view der typecaster data fra ext tabellen | *USE cars\_template;*  *DROP VIEW IF EXISTS `cars\_stage`.`tfm\_vw\_american\_cars`;*  *CALL `cars\_template`.`sp\_create\_tfm\_view`('cars\_stage', 'american\_cars');* |
| check at viewet er designet som du forventer  Afprøv viewet med  SELECT \* FROM `cars\_stage`.`tfm\_vw\_american\_cars` | I dette tilfælde vises create staementen i en lang tekststren på 1. linje    Der findes en beutifier feature blandt kommandoerne i SQL editor vinduet |
| **1.4.3 sp\_create\_tfm\_sp**  Kommandoen danner en sp, der trækker data fra ext tabellen gennem tfm viewet og lander dem i dat tabellen | USE cars\_template;  DROP PROCEDURE IF EXISTS `cars\_stage`.`tfm\_sp\_american\_cars`;  CALL `cars\_template`.`sp\_create\_tfm\_sp`('cars\_stage', 'american\_cars'); |
| I dette tilfælde får vi denne fejlkode:  Error Code: 1295 This command is not supported in the prepared statement protocol yet  Jeg lærte at man ikke kan danne sp’er gennem sp’ere i MySql. |  |
| Dette er en major showstopper ift. at bruge sp’ere som halv automatiserings grundlag. Vi kunne måske komme omkring det på mange måder, vi taler trods alt om et operatør drevent ”API”.  Vi fortsætter og anvender en workaroud, der også viser dig hvordan du debigger store procedures.  Bemærk den gule pil i skærmbilledet ovenfor. I dette tilfælde selectere vi hhvad der står i vores dynamiske sql tekststreng. | |
| Peg på resultatet I ‘Result Grid’ højreklik og i kontekst menuen vælg ’Copy Field’  Paste indholdet til et nyt SQL Command vindue |  |
| Fjern ping fra start og slut, og du har kommandoen klar, du nu selv må eksekvere. |  |
| Check cars\_stage.tfm\_sp\_american\_cars definition |  |

# Konklusion om replika et BI udviklingsflow

Der mangler mange ting endnu før vi kan sige at vi har det fulde featuresæt som er nødvendigt i et produktionsklar BI rådata lag.

* I alle sp’er der afvikles automatiseret, skal der laves try- catch og logges i en fejldatabase, måske er dette et domæne helt for sig selv, og kalder på oprettelse af en egentlig system database
* Vi har slet ikke favnet batch begrebet, som er nødvendigt når vi regelmæssigt importere data. Formålet er at danne historiske data, og det når vi ikke i denne replika, der kun håndtere en enkelt import.
* Historiske data dannes typisk i en merge mellem dat tabellen og vores eksisterende billede af hvad vi fandt sidste gang vi aflæste on-line.

# Afprøvning: Rådata ETL

Vi nåede endeligt til at skulle afprøve vores transform load.

|  |  |
| --- | --- |
| **2. Transform**  Kommandoen kører en extract, transform, load som den ville blive eksekveret automatisk | *call cars\_stage.tfm\_sp\_american\_cars()* |
| Check data i tabellen  Cars\_stage.dat\_american\_cars | *SELECT \* FROM cars\_stage.dat\_american\_cars;* |
| Og dermed blev vi færdige med denne del af vores mainflow |  |