

# Kode review - Malte Hviid-Magnussen

<https://github.com/MalteMagnussen/LegoHouse>

Foretaget d. 27/03-2019 af [Rúni Vedel Niclasen](#). Reviewet er af [Version 251d6cc5fd](#)

## Clean-build / Github

Clean & build → Run var en succes.

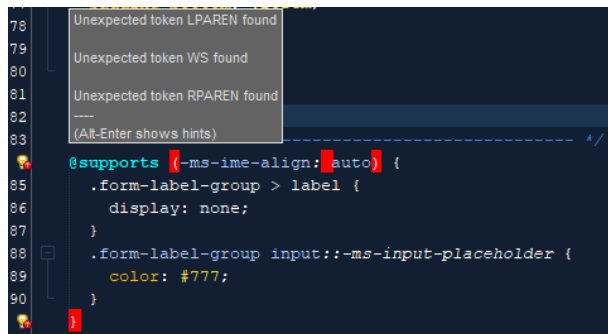
- Fejl i floating-labels.css

+ Flot github struktur og readme.

+ JavaDocs - også online

+ Deployment

+ SQL DDL/DML script - DB hedder 'useradmin'?



## Grøn

1. Det færdige system skal være struktureret 3 lags modellen - præsentationslag, forretningslag og datalag.
  - Godkendt. Dog starter strukturen med en folder der hedder **malte(?)**
  - Nogle af JSP filerne ligger i **Web Pages/WEB-INF**, andre ligger i **/Web Pages**.
2. Det forventes at der er en facade mellem præsentations og forretningslag
  - Godkendt
3. Det forventes at der IKKE kaldes forretningslogik (eller datamappers) fra JSP siderne.
  - Godkendt
4. Det forventes at session og requests attributter anvendes korrekt.
  - Request parametre anvendes korrekt.
5. Vi vil anse det for en fejl hvis der bruges redirect hvor der burde bruges forward.
  - Der gøres brug af den udleverede FrontController som anvender requestDispatcher.
6. Det forventes at styklisterne ikke gemmes i databasen, men beregnes i forretningslaget.
  - Godkendt
7. Du skal håndtere exceptions i data laget - ikke blot catch og så skrive stack trace ud på System.out
  - Der er implementeret to custom exceptions, begge bliver fanget af FrontController.

## Gul

1. Du skal lave en specialiseret exception som du kaster fra datalaget og fanger i servlet.
  - Godkendt
2. JSP siderne må kun hente information via request og servlet
  - Kører ofte på session, dog på servlet.

## Rød

1. Det forventes at du bruger en front-controller som vist i ugens oplæg.
  - Godkendt

## Gemmengang af koden

+ Dokumentation

### Java:

- + switch (origin) - fed, anderledes made at gøre det på.
- + Fanger yderligheder, f.eks. tages der højde for forkert brugerinput i Product.java, selvom .jsp siden allerede håndterer dette.
- + Above and beyond med synchronization & serialization.
- + Programmet virker meget kompakt i form af at der kun er 3-4 commands.
- + Gennemtænkte exceptions – en til index, en til role-page

+/- Brug af *session* visse steder i programmet (Bl.a. *Product.java*) kan der ses nærmere på. Kan *request* ved fordel bruges i stedet? Er det med vilje, sådan som klassen er bygget op?

```
151     private void orderlogic(User user, int id, HttpSession session)
152     {
153         List<Order> orders = user.getOrders();
154         for (Order order : orders)
155         {
156             if (order.getId() == id)
157             {
158                 session.setAttribute("order", order);
159                 ControllerFacade c = ControllerFacadeImpl.getInstance();
160                 BOM bom = c.getBOM(order);
161                 session.setAttribute("BOM", bom);
162             }
163         }
164     }
```

÷ Brug af *ex.GetMessage* ved visse exceptions viser potentielt forvirrende besked til brugeren  
Redirect.java - Sjov, men virker til hvad den skal.

### JSP:

- +/- Visse stedet med in-document styling, når der nu er en .css fil.
- + Brug af EL/ JSTL
- + Den kompakte struktur fra java-delen følger her og det er programmør-venligt.

## Gemmengang af hjemmesiden

+ Den kompakte struktur fra java-delen følger her og det giver et godt *flow*. Yder en god brugervenlighed.

Session er ret striks og kører fortsat selvom et andet login fejler:

(1. Log ind som robin, 2. gå tilbage og prøv med en anden bruger. Robin er stadig på session og det vises øverst)



### Employee:

Drop-down menuen 'Shop' returnerer en employee til employee-page og ikke shop.

### Customer:

Ikke mulighed for at vælge hvorvidt du vil have vinduer, døre, forbandt. 1 algoritme. (Ikke påkrævet)

Flot Order-table, både for customer og employee

Flot stykliste - Den er bare på dansk, når resten af siden er engelsk.

God funktionalitet, det håndhæver funktionalitet og enkelthed, brugervenlighed og de ideer vi omtalte til peer-review.