IT-DIPLOMUDDANNELSEN

DATABASESYSTEMER OG WEB

Eksamensprojekt

af

Jan Schrøder Hansen

eFTERÅR 2010

**INDHOLD**

[1. Indledning 3](#_Toc278549472)

[2. Opgavebeskrivelse 3](#_Toc278549473)

[3. ER Diagram 3](#_Toc278549474)

[4. RI Diagram 6](#_Toc278549475)

[5. Normalisering 7](#_Toc278549476)

[6. Egenskabstabel 7](#_Toc278549477)

[6.1. Tabeloprettelse 9](#_Toc278549478)

[7. SQL udtræk og views 9](#_Toc278549479)

[8. Indeksering 10](#_Toc278549480)

[9. Sikkerhed og brugere 11](#_Toc278549481)

[10. Konklusion 11](#_Toc278549482)

[11. Bilag 13](#_Toc278549483)

[11.1. Brugervejledning 13](#_Toc278549484)

[11.2. Udviklingsmiljø 19](#_Toc278549485)

[11.3. Kode 20](#_Toc278549486)

[11.3.1. dk.jsh.itdiplom.dbsw.bari.domain 20](#_Toc278549487)

[11.3.2. dk.jsh.itdiplom.dbsw.bari.business 29](#_Toc278549488)

[11.3.3. dk.jsh.itdiplom.dbsw.wicket 32](#_Toc278549489)

[11.3.4. Hibernate configuration 54](#_Toc278549490)

[11.4. SQL til oprettelse og initering af databasen 54](#_Toc278549491)

[11.5. Indhold på vedlagte CD 57](#_Toc278549492)

# Indledning

Dette eksamensprojekt er lavet i forbindelse med faget Web og serverprogrammering på IT- Diplomuddannelsen, Ingeniørhøjskolen i København.

Faget har taget udgangspunkt i bøgerne ”Database Management Systems” af Raghu Ramakrishnan og Johannes Gehrke og ”SQL i praksis” af Ben Forta. Samt materiale udarbejdet af underviser Torben Lund.

# Opgavebeskrivelse

For at komme igennem så meget af materialet i faget som muligt, har jeg valgt at lave en webløsning til håndtering af ønsker og fejlreporter, til et eller flere softwareprodukter.

Der skal være mulighed for at følge status på fejl og ønsker. Er fejlen eller ønsket godkendt, afvist, er det under udvikling, under test etc. Derudover skal der være mulighed for at uploade forskellige filer såsom skærmdumps af fejl, eller prototyper på ny skærmlayouts og andre filer som kan hjælpe til at belyse en sag.

Programmets navn er BaRI, som står for ”Bugs and Request Interceptor”.

Det er en løsning, jeg startede på under faget Web og Serverprogrammering, efteråret 2009, rapporten kan findes på den vedlagte cd. I denne opgave var mit fokus på et web framework kaldet Wicket[[1]](#footnote-2) . En lille del af opgaven skulle også demonstrere at løsningen kunne persistere data i en database. Så der blev lavet 2 tabeller. I denne opgave vil jeg forsætte på denne løsning, men med fokus på database delen.

Programmet udvikles i sproget Java[[2]](#footnote-3). Med Java følger databasen JavaDB[[3]](#footnote-4), derudover benyttes Hibernate[[4]](#footnote-5) som bro mellem den relationelle database verden og den objekt orienteret verden.

I de følgende afsnit, gennemgås de forskellige designe faser, som benyttes i forbindelse med design af en database. Startende med et *entity-relationship* digram[[5]](#footnote-6), herefter kaldet et ER diagram.

# ER Diagram

Følgende figur er et ER diagram bestående af entiteter (kasser), som er identificerbare selvstændige objekter, navngives ofte med et navneord. Samt relationer (kasser på spidsen) som beskriver relationer mellem entiteter, navngives ofte med udsagnsord. Og sidst men ikke mindst af attributter (cirkler), som kan knyttes til både entiteter og relationer. Attributter beskriver enkelte data som navn, adresse etc. På den eller de attributter, som entydig identificere en relation, er navnet på attributten understreget.

ER Diagram.wmf

Figur 1- ER Diagram

Som det fremgår af ovenstående figur, så har alle entiteter en nøgle kaldet id (alle de gule attributter). Dette skyldes at jeg har valgt at benytte frameworket Hibernate, som er nemmere at arbejde med, når der bare benyttes et fortløbende tal som nøgle.

Som det fremgår af diagrammet er der et Produkt, som indeholder navne på de produkter som systemet skal kunne modtage fejl og ønsker til. Så er der en BariUser, som indeholder de brugere som har adgang til systemet. Mellem Produkt og BariUser er der en mange til mange relation kaldet userGroup som beskriver hvilke produkter de enkelte brugere har adgang til. En bruger har også en rolle, se userRole, som beskriver om en bruger er en udvikler, administrator eller en alm. bruger. En administrator kan alt i systemet, og som den eneste kan denne oprette nye brugere og produkter. En alm. bruger kan oprette fejl og ønsker, samt oprette diskussionsindlæg og uploade skærmdumps og andet. En udvikler kan det samme som en alm. bruger, samt ændre på en udviklingsstatus, som siger noget om hvor langt et ønske/fejl er mht. udvikling og test.

Centralt i systemet er der en BariCase som beskriver en fejl eller et ønske, til hvert BariCase kan der knyttes flere diskussionsindlæg (DiscussionMessage) og igen til hvert diskussionsindlæg kan der igen knytte flere filer (Attachement), som kan være skærmdumps eller andet, som kan benyttes til at uddybe et indlæg.

En BariCase kan være af type fejl eller ønske (Type), samt have en sagsstatus (CaseStatus), som kan være ”Ny”, ”Behandles”, ”Godkent”, ”Afvist” eller ”Afsluttet”, samt en udviklerstatus (DevStatus ) som kan være ”Ej begyndt”, ”I gang”, ”Klar til test”, ”Testet” og ”I produktion”.

ER diagrammet som sådan, kan ikke direkte implementeres i en database, så næste design fase, er et *Referential integrity*[[6]](#footnote-7)diagram, herefter kaldet RI Diagram.

# RI Diagram

Et RI diagram benyttes til at beskrive de referentielle sammenhænge der skal være mellem tabellerne i en database. En database benytter referentiel integritet til at sikre, at data er konsistente. F. eks. at der ikke slettes rækker i en tabel, som der refereres til fra en anden tabel.

RI Diagram.wmf

Figur 2 - RI diagram

Som det fremgår af ovenstående RI diagram, så mappes næste alle entiteter fra ER diagrammet, over til en relation/tabel i RI diagrammet. Der er en ekstra relation som er UserGroup. I ER diagrammet var UserGroup en mange til mange relation. Og det mappes til en tabel i RI diagrammet. Denne tabel har så en ”1 en til mange” relation til Product og tilsvarende til BariUser. Alle de andre relationer fra ER diagrammet, beskrives har med 1 til mange relationer og fremme nøgler (FK – Foreign Key). F. eks. BariUser har en FK til UserRole. Derudover har alle tabeller en primær nøgle (PK), som alle er understreget i diagrammet. I ovenstående diagram kan også ses hvilke felter, som kan være tomme (null), de har en åben cirkel for felt navnet. F.eks. i BariCase, er der et felt som hedder conclusion. Som er en konklusion på en fejl eller et ønske. Og det kan i sagens natur først udfyldes til allersidst.

Efterfølgende skal ovenstående RI diagram efterprøves for nogle normaliserings regler.

# Normalisering

Normalisering er en form for database kvalitetssikring, som sikre et database design overholder visse regler. Bl.a. at samme data kun skrives en gang. Følgende er en kort beskrivelse af de 3 første normalforme.

**1. Normalform:**

* Tabeller skal have en primærnøgle
* Der må ikke være repeterende grupper.

**Alle tabeller har en primærnøgle som hedder id.**

**2. Normal form:**

* Tabellen skal være på første normal form.
* Man må ikke kunne udlede værdien af et felt uden for primærnøglen, af en del af nøglen.

**Alle tabeller har en primærnøgle på et felt.**

**3. Normal form:**

- Tabellen skal være på anden normal form.

- Man må ikke kunne udlede værdien af et felt uden for primærnøglen af et andet felt (eller en kandidatnøgle). F. eks. postnr. og by. Hvor by kan udledes af postnr.

**Der er ingen felter som kan udledes af andre (ikke nøgle) felter.**

Ovenstående RI diagram fra afsnit 4, kan hermed siges at være det endelige design. Efterfølgende skal de enkelte tabeller specificeres men hensyn til datatyper, størrelser, constraints, samt uddybende kommentere til de enkelte felter.

# Egenskabstabel

Da jeg har valgt at bruge JavaDB som database, vil det være datatype, som benyttes i denne database, som bruges i det følgende.

Fælles for alle mine tabeller er at de har et primærnøgle kaldet ”id”. Denne er af typen bigint, og det er en nøgle som database selv generere, når der oprettes en ny række. Derudover har alle tabeller et felt kaldet ”version”.

Version er ikke beskrevet tidligere, da det er Hibernate specifikt. Det bruges af Hibernate til optimistisk låsning[[7]](#footnote-8), som i korte træk består i, at systemet går ud fra, at de enkelte brugere ikke arbejder på samme data. Hvis der så er konflikter, så er det først til møllen princippet, der bestemmer, hvem der kommer af med sin opdatering.

F. eks. hvis bruger A læser en række med id = 1 og version = 1, og en bruger B læser den samme række. Bruger A opdatere rækken, hvor Hibernate øger version med 1, så den nu er 2. Nu vil bruger B også opdatere denne række. Men bruger B får en fejl. Fordi Hibernate prøver at lave en update med følgende where sætning: ”where id = 1 and version = 1”. Da denne række ikke findes mere, vil Hibernate returnere en exception, som fanges af systemet og fortælle brugeren at de data, han forsøgte at gemme, er rettet af en anden bruger i mellemtiden.

For at afgrænse opgaven, har jeg valgt ikke at implementere følgende tabeller: UserRole, Type, CaseStatus, DevStatus samt Attachment.

UserRole, Type, CaseStatus og DevStatus er alle ”type” tabeller, dem har jeg valgt at implementer som java enums i stedet for. Det betyder at jeg har erstattet alle de fremmenøgler som pegede på disse, med varchars. Disse varchars har så fået nogle constraints, som siger hvilke lovlige tekster der kan skrives i disse felter. F.eks. for Type gælder at det kun er lovligt at bruge ”ERROR” og ”REQUEST”.

I det følgende vil de enkelte tabeller som implementeres, beskrives felt for felt, dog uden id og version, som er beskrevet tidligere og er fælles for alle. Null constraint undlades også, da det kun er 2 felter som kan være null. Og det er finished og conclusion i tabellen BariCase.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tabel** | **Feltnavn** | **Datatype** | **Forklaring** | **Domain** | **Unique** |
| **Product** | name | varchar(50) | Produktnavn |  | Ja |
| **UserGroup** | bariUser\_id | bigint | FK til bariUser |  | Ja, sammen med product\_id |
|  | product\_id | bigint | FK til Procuct |  | Ja, sammen med bariUser\_id |
| **BariUser** | fullname | varchar(50) | Brugers fulde navn |  | Nej |
|  | login | varchar(20) | Bruges login kode |  | Ja |
|  | password | varchar(20) |  | Mindst 3 tegn | Nej |
|  | userRole | varchar(10) | Brugers rolle | En af følgende værdier: ADMIN, DEVELOPER, NORMAL | Nej |
| **BariCase** | title | varchar(50) | Overskrift til sagen. |  | Nej |
|  | created | timestamp | Dato/tid for oprettelse af sagen. |  | Nej |
|  | finished | timestamp | Dato/tid for afslutning af sagen. |  | Nej |
|  | description | varchar(400) | Beskrivelse af sagen. |  | Nej |
|  | conclusion | varchar(400) | Konklusion på sagen. |  | Nej |
|  | product\_id | bigint | FK id til Product tabellen. Til det produkt som sagen omhandler. |  | Nej |
|  | bariUser\_id | bigint | FK id til BariUser tabellen. Til den bruger som har oprettet sagen. |  | Nej |
|  | type | varchar(10) | Type som beskriver om sagen omhandler en fejl eller et ønske. | En af følgende værdier: ERROR, REQUEST | Nej |
|  | caseStatus | varchar(15) | Sags status. | En af følgende værdier: NEW, CONSIDERING, APPROVED, REJECTED, DONE | Nej |
|  | devStatus | varchar(15) | Udvikler status. | En af følgende værdier: NOTSTARTED, STARTED, READYTOTEST, TESTED, INPRODUCTION | Nej |
| **DiscussionMessage** | created | timestamp | Dato/tid for oprettelse af diskussionsindlægget. |  | Nej |
|  | message | varchar(400) | Selve indlægget. |  | Nej |
|  | bariCase\_id | bigint | FK id til tabellen BariCase. Til den sag som dette indlæg omhandler |  | Nej |
|  | bariUser\_id | bigint | FK id til tabellen BariUser. Til den bruger som har oprettet indlægget. |  | Nej |

## Tabeloprettelse

Selve oprettelsen af tabellerne, sker vha. af Hibernate. Hibernate har værktøjer til at danne java klasser ud fra et databaseskema/ddl, og omvendt at skabe tabeller udfra java klasser med annotations. Jeg har valgt den sidste mulighed. De javaklasser som er brugt findes i pakken dk.jsh.itdiplom.dbsw.domain pakken (se under bilag afsnit 11.3.1), og resultatet kan ses af den skemafilen bari\_ddl.sql som oprettes. Se bilag afsnit 11.4. Derudover har jeg filen bari\_init.sql som jeg vedligeholder manuel, som indeholder enklete constraings, jeg ikke har kunnet lavet i java klasserne, samt enkelte system data, såsom 3 brugere og 3 produkter. Se bilag afsnit 11.4.

# SQL udtræk og views

Jeg har valgt ikke at bruge nogle views direkte i programmet, men at benytte mig af Hibernates Query Language (herefter bare HQL). HQL er i korte træk bare SQL hvor man benytter javaklasser i stedet for database tabeller. Et eksempel fra koden:

select bariCase

from dk.jsh.itdiplom.dbsw.bari.domain.BariCase bariCase

where bariCase.type = :type and product.id = :productid;

Som det ses, kan HQL returnere hele objekter af javaklasser, her BariCase. Det betyder at man slipper for at mappe fra en tabel til et objekt. Dette gøres af Hibernate. ”:type” og ”:productid” er variabler som indsættes på kørselstidspunktet. Det er også muligt at benytte alm. SQL vha. Hibernate, fordi Hibernate gør brug af en JDBC[[8]](#footnote-9) driver. Flere eksempler på HQL kan ses i koden i pakken dk.jsh.itdiplom.dbsw.business. Se under bilag afsnit 11.3.2.

Som eksempel på SQL sætninger, som kunne oprettes som views (create view <navn>), har jeg lavet nogle statistik udtræk.

|  |
| --- |
| Find antallet af sager som har en given status. De skal grupperes efter type (ERROR, REQUEST) og status (NEW, CONSIDERING etc.). |
| select type, casestatus, count(\*) as "no of cases"  from baricase  group by type, casestatus; |
| Find antallet af diskussionsindlæg pr. sag. Grupperes efter type, titel. Antal udskrives i faldende orden. |
| select type, title, count(baricase\_id) as "No of discussion per case"  from baricase  left outer join discussionmessage on (baricase.id = discussionmessage.baricase\_id)  group by type, title, baricase\_id  order by count(baricase\_id) desc; |
| Find antallet af sager med en given kombination af type, status og bruger. |
| select bariuser.fullname, baricase.type, baricase.casestatus, count(baricase.bariuser\_id) as "No of cases"  from baricase  inner join bariuser on (baricase.bariuser\_id = bariuser.id)  group by bariuser.fullname, baricase.type, baricase.casestatus; |

# Indeksering

Der er pt. ikke oprettet indekser. Men gode kandidater til felter som kunne indekseres, er fremmenøgler (FK), samt felter i som indgår i joins og where betingelser. I BaRI er der kun tre tabeller, som kan vokse anseligt, det er BariCase, DiscussionMessage og evt. Attachment. Størrelsen på andre tabeller som f.eks. Product og BariUser, må anses for at være beskeden.

Så en god kandidat kunne være bariCase\_id (FK til BariCase) i DiscussionMessage, så det er hurtigt at finde alle diskussionsindlæg til en sag. Og det samme vil gælde for discussionMessage\_Id i Attachment, for hurtigt at finde attachemets til et diskussionsindlæg.

Hvis der mod forventning skulle komme tusindvis af bruger på systemet, så kunne der laves et index på feltet login i BariUser, for at finde en bruger lidt hurtigere i login processen.

Men min holdning er at der først skal oprettes indeks, når behovet opstår. Det betyder at man løbende skal holde øje med systemet, for at se om der er nogle SQL udtræk som begynder at tage langt tid. Dette kan gøres vha. logging, Hvor tiden for kritiske SQL udtræk logges. Men kan også lave en query execution plan[[9]](#footnote-10), som kan fortæller hvordan databasen udfører en givent SQL udtræk. En execution plan siger noget om der benyttes index, eller om hele tabeller skannes igennem, samt tidsforbrug, IO resurser, CPU resurser m.m.

Indeks oprettes med ”create index on <table> (<column names>)”

# Sikkerhed og brugere

Der er kun oprettet en database bruger, kaldet bari. Som benyttes som database administrator og som også benyttes af web applikationen. Dette er ikke optimalt, som minimum bør der være en applikationsbruger, som bruges af web-applikationen, en database administrator, samt evt. en bruger som har læseadgang til diverse udtræk.

I selve applicktionen opereres der med 3 type bruger, En administrator, alm. bruger og en udvikler. Følgende er en tabeloversig over de rettigheder, som den enkelte bruger har i forhold til de enkelte tabeller, også kaldet en tilgangsmatrix. Alle disse adgange styres af selve webapplikationen.

|  |  |  |  |
| --- | --- | --- | --- |
| Bruger type  Tabel | Administrator  (ADMIN) | Udvikler  (DEVELOPER) | Alm. bruger  (NORMAL) |
| Product | S I U D | S | S |
| UserGroup | S I U D | S | S |
| BariUser | S I U D | S | S |
| BariCase | S I U D | S I U D | S I U D[[10]](#footnote-11) |
| DiscussionMessage | S I U D | S I | S I |

De enkelte koder beskrives i følgende tabel.

|  |  |
| --- | --- |
| Symbol | Forklaring |
| S | Select |
| I | Insert |
| U | Update |
| D | Delete |

Hvis en BariCase slettes, så slettes alle tilhørende DescussionMessage.

# Konklusion

Det har været en god proces, startende med et ER-diagram, mapping til et RI-diagram, normalisering samt endelige database design, med felttyper, constrains m.m.

Jeg er ikke ramt ind i nogen forhindringer eller overraskelser i forløbet, og det skyldes helt klart mit valg af Hibernate Dette valg har haft stor indflydelse på designet, det medførte bl.a. at alle tabeller fik en priæmrnøgle, som bare er et fortløbende nummer, genereret af databasen. Det har betydet at jeg ikke skulle bruge kræfter på at finde entydige primærnøgler, som igen har betyder at de 2 første normaliseringsregler er opfyldt.

Personlig have jeg nok valgt at starte ud med Use Case[[11]](#footnote-12) beskrivelser, samt et logisk UML[[12]](#footnote-13) klassediagram. Klassediagramme kunne så mappes til et RI-diagram, hvor der tages stilling til hvordan arv, klasser imellem, håndteres i en database sammenhæng, om der er nogle mange til mange relationer, som skal indføres. Da mange til mange relationer ikke er noget specielt problem, i en object orientert sammenhæng.

Når man så havde en RI-Diagram, så kunne man følge fremgangsmåden som også er benyttet i dette projekt. Dvs. gennemgå RI-diagrammet efter normaliseringsreglerne, lave en egenskabsstable, etc.

# Bilag

## Brugervejledning

Følgende er en kort gennemgang af de enkelte sider i BaRI applikationen. Det er pt. kun muligt at oprette, rette og slette sager. Samt tilføje kommentarer til disse. Administration af brugerer og applikationer foregår direkte i databasen af den database administrator.

Det første en bruger møder er en login side.

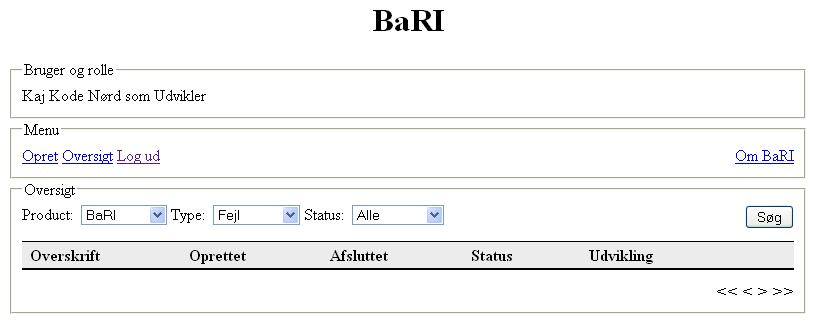


Figur 3 - Login side

En login side med fejl i login eller password.

*Figur 4 - Login side med fejl*

Hvis det er et system uden nogen sager, så kommer man til en tom oversigt, efter login.

*Figur 5 - Tom oversigt*

Her ses det at det er ”Kaj Kode Nørd” som er indlogget som Udvikler.

Der er en Menu, som findes på alle sider. Fra denne kan man oprette en ny sag, gå til denne oversigt, gå til en side om BaRI, samt logge ud. Log ud retunere til Login siden.

På denne side, skal man vælge et Produkt, samt type (Fejl eller ønske) og evt. en sagsstatus. Og efter en tryk på ”Søg” knappen vil oversigen vise en liste med de sager som opfylder søgekriterierne.

For at oprette den første sag, trykkes der på ”Opret” linket i menuen, og følgende side vises.



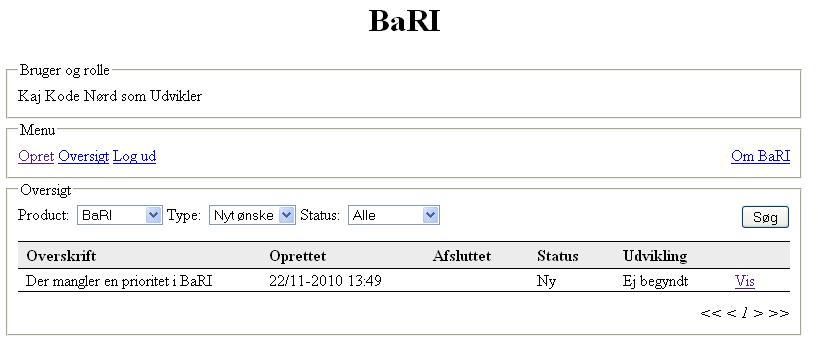
Figur 6 - Opret ny sag

Tryk på knappen ”Gem”, gemmer indtastede data og returnerer til oversigt siden.

Følgende er et eksempel på felter som skal udfyldes, men mangler.

*Figur 7 - Forsøg på at oprette en ny sag, hvor der mangler obligatoriske felter*

Tilbage til oversigt siden, nu med en enkelt sag.

*Figur 8 - Oversigt nu men en sag*

” << < 1 > >>” under listen betyder at der er en side med data som opfylder søgekriteriet.

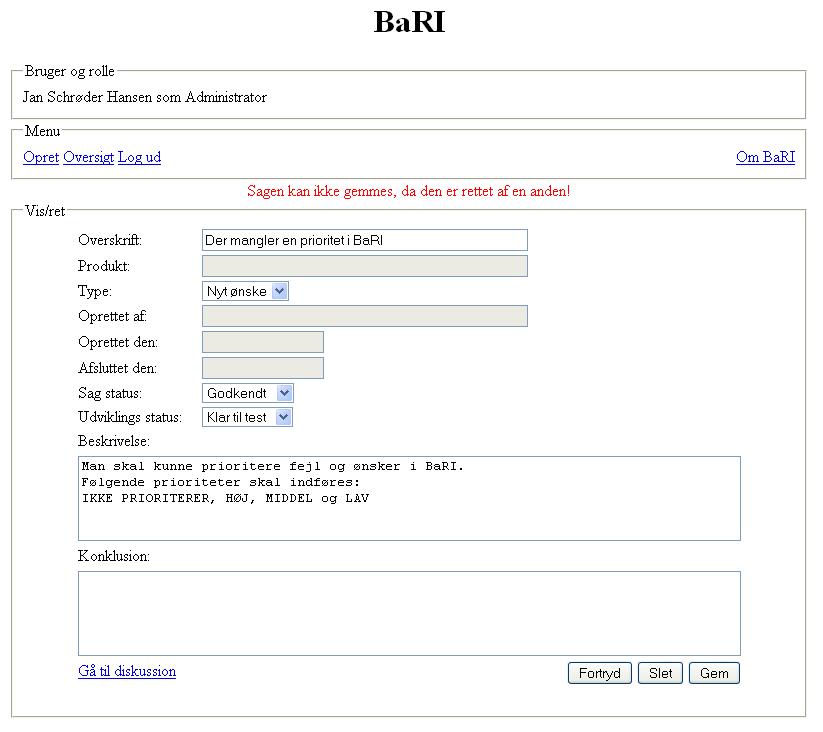
”<<” er et link til første side, ”<” går en side tilbage, ”>” går en side frem og ”>>” går til sidste side.

Hvis man trykker på ”Vis” linket så kommer man til en opdaterings side for sagen.

*Figur 9 - Opdaterings side*

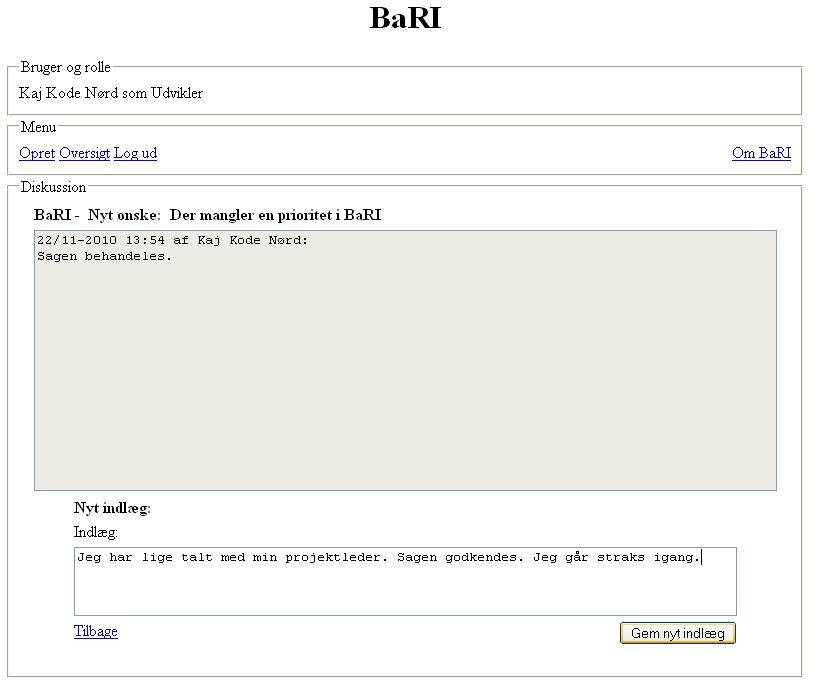
Grå felter kan ikke rettes. De hvide felter kan rettes ved at trykke på ”Gem” knappen. En sag kan slettes ved at trykke på ”Slet” knappen. Hvorefter der fremkommer en ”Er du sikker” dialog.

Følgende er et eksempel på en ”optimistisk låsning” fejl. Dvs. at de på siden viste data, er retten af en anden person i mellemtiden.



Figur 10 - Optimistisk låsning eksempel

For at se diskussionsbeskeder og for at oprette disse, trykkes der på ”Gå til diskussion” linket, som viser følgende side.



Figur 11 - Diskussion side

Her kan der oprettes nye diskussionsindlæg, ved at skrive i ”Indlæg” boksen og trykke på ”Gem nyt indlæg” knappen. Alle tidligere indlæg ses i den grå boks, med dato/tid og bruger.

Den sidste side er ”Om BaRI” siden:

*Figur 12 - Op BaRI siden*

## Udviklingsmiljø

Den kode som er vedlagt på den vedlagte cd, er lavet vha. af et Java IDE (udviklingsværktøj) kaldet NetBeans[[13]](#footnote-14) version 6.8. Java versionen skal være en 1.6 eller højdere, samt den JavaDB som er en del af denne.

Jeg har været inde på de java frameworks der benyttes, som er følgende:

* Hibernate – version 3.2
* Wicket – version 1.4

Web serveren, som har været brugt til test er en Apache Tomcat[[14]](#footnote-15) version 6. Det hele er testet inde fra Netbeans, som har mulighed for at starte en JavaDB server, samt en Tomcat server.

Al kode er gemt i et SubVersion[[15]](#footnote-16) repository, som bestyres af google, via deres GoogleCode[[16]](#footnote-17) service. Her har jeg oprettet følgende hjemmeside til projektet.

<http://code.google.com/p/bari/>

Her står der meget kort om projektet, samt det er muligt at downloade koden. Der er sjovt nok også et lille ”Issues” side/fane, hvor man kan ind-rapportere fejl til projektet.

## Kode

Koden er delt op de enkelte i lag/områder vha. java pakker (package). Se følgende figur.

pakkediagram.wmf

Figur 13 - Pakkr diagram

De grønne pakker er udviklet i forbindelse med dette program. De 2 lyseblå pakker henviser til de 2 java frameworks som benyttes.

De grønne pakker kan også ses som min version af Model-View-Controller[[17]](#footnote-18) mønstret (MVD). Hvor wicket pakken er view, business er controller og domain er model.

I det følgende listes koden fra de 3 grønne pakker.

### dk.jsh.itdiplom.dbsw.bari.domain

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\domain\BariCase.java |

1 package dk.jsh.itdiplom.dbsw.bari.domain;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.CaseStatus;

4 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.DevStatus;

5 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.Type;

6 import java.io.Serializable;

7 import java.util.Date;

8 import javax.persistence.\*;

9

10 /\*\*

11 \* BariCase entity class.

12 \*

13 \* @author Jan S. Hansen

14 \*/

15 @Entity

16 public class BariCase implements Serializable {

17 private static final long serialVersionUID = 1L;

18

19 @Id

20 @GeneratedValue(strategy = GenerationType.IDENTITY)

21 protected Long id;

22 @Version

23 @Column(nullable = false)

24 protected Integer version;

25 @Column(length=50, nullable = false)

26 protected String title;

27 @Enumerated(EnumType.STRING)

28 @Column(length=10, nullable = false)

29 protected Type type;

30 @ManyToOne(optional=false)

31 @org.hibernate.annotations.ForeignKey(name="fk\_from\_baricase\_to\_bariuser")

32 protected BariUser bariUser;

33 @ManyToOne(optional=false)

34 @org.hibernate.annotations.ForeignKey(name="fk\_from\_baricase\_to\_product")

35 protected Product product;

36 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

37 @Column(nullable = false)

38 protected Date created;

39 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

40 protected Date finished;

41 @Enumerated(EnumType.STRING)

42 @Column(length=15, nullable = false)

43 protected CaseStatus caseStatus;

44 @Enumerated(EnumType.STRING)

45 @Column(length=15, nullable = false)

46 protected DevStatus devStatus;

47 @Column(length=400, nullable = false)

48 protected String description;

49 @Column(length=400)

50 protected String conclusion;

51

52 public BariCase() {

53 }

54

55 public BariCase(String title, Type type, BariUser bariUser,

56 Product product, Date created,

57 Date finished, CaseStatus caseStatus, DevStatus devStatus,

58 String description, String conclusion) {

59 this.title = title;

60 this.type = type;

61 this.bariUser = bariUser;

62 this.product = product;

63 this.created = created;

64 this.finished = finished;

65 this.caseStatus = caseStatus;

66 this.devStatus = devStatus;

67 this.description = description;

68 this.conclusion = conclusion;

69 }

70

71 public Long getId() {

72 return id;

73 }

74

75 public void setId(Long id) {

76 this.id = id;

77 }

78

79 public Integer getVersion() {

80 return version;

81 }

82

83 public String getTitle() {

84 return title;

85 }

86

87 public void setTitle(String title) {

88 this.title = title;

89 }

90

91 public Type getType() {

92 return type;

93 }

94

95 public void setType(Type type) {

96 this.type = type;

97 }

98

99 public BariUser getBariUser() {

100 return bariUser;

101 }

102

103 public void setBariUser(BariUser bariUser) {

104 this.bariUser = bariUser;

105 }

106

107 public Product getProduct() {

108 return product;

109 }

110

111 public void setProduct(Product product) {

112 this.product = product;

113 }

114

115 public Date getCreated() {

116 return created;

117 }

118

119 public void setCreated(Date created) {

120 this.created = created;

121 }

122

123 public Date getFinished() {

124 return finished;

125 }

126

127 public void setFinished(Date finished) {

128 this.finished = finished;

129 }

130

131 public CaseStatus getCaseStatus() {

132 return caseStatus;

133 }

134

135 public void setCaseStatus(CaseStatus caseStatus) {

136 this.caseStatus = caseStatus;

137 }

138

139 public DevStatus getDevStatus() {

140 return devStatus;

141 }

142

143 public void setDevStatus(DevStatus devStatus) {

144 this.devStatus = devStatus;

145 }

146

147 public String getDescription() {

148 return description;

149 }

150

151 public void setDescription(String description) {

152 this.description = description;

153 }

154

155 public String getConclusion() {

156 return conclusion;

157 }

158

159 public void setConclusion(String conclusion) {

160 this.conclusion = conclusion;

161 }

162 }

163

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\domain\BariUser.java |

1 package dk.jsh.itdiplom.dbsw.bari.domain;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.UserRole;

4 import java.io.Serializable;

5 import javax.persistence.\*;

6

7 /\*\*

8 \* BariUser entity class.

9 \*

10 \* @author Jan S. Hansen

11 \*/

12 @Entity

13 public class BariUser implements Serializable {

14 private static final long serialVersionUID = 1L;

15

16 @Id

17 @GeneratedValue(strategy = GenerationType.IDENTITY)

18 protected Long id;

19 @Version

20 @Column(nullable = false)

21 protected Integer version;

22 @Column(length=20, nullable = false, unique=true)

23 protected String login;

24 @Column(length=20, nullable = false)

25 protected String password;

26 @Column(length=50, nullable = false)

27 protected String fullname;

28 @Enumerated(EnumType.STRING)

29 @Column(length=10, nullable = false)

30 protected UserRole userRole;

31

32 public BariUser() {

33 }

34

35 public BariUser(String login, String password, String fullname,

36 UserRole userRole) {

37 this.login = login;

38 this.password = password;

39 this.fullname = fullname;

40 this.userRole = userRole;

41 }

42

43 public Long getId() {

44 return id;

45 }

46

47 public void setId(Long id) {

48 this.id = id;

49 }

50

51 public Integer getVersion() {

52 return version;

53 }

54

55 public String getFullname() {

56 return fullname;

57 }

58

59 public void setFullname(String fullname) {

60 this.fullname = fullname;

61 }

62

63 public String getLogin() {

64 return login;

65 }

66

67 public void setLogin(String login) {

68 this.login = login;

69 }

70

71 public String getPassword() {

72 return password;

73 }

74

75 public void setPassword(String password) {

76 this.password = password;

77 }

78

79 public UserRole getUserRole() {

80 return userRole;

81 }

82

83 public void setUserRole(UserRole userRole) {

84 this.userRole = userRole;

85 }

86 }

87

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\domain\Constants.java |

1 package dk.jsh.itdiplom.dbsw.bari.domain;

2

3 import java.util.ArrayList;

4 import java.util.List;

5

6 /\*\*

7 \* Constants and enums.

8 \*

9 \* @author Jan S. Hansen

10 \*/

11 public class Constants {

12 private Constants() {}

13

14 /\*\*

15 \* BaRI case type: REQUEST, ERROR.

16 \*/

17 public enum Type {

18 REQUEST("Nyt ønske"),

19 ERROR("Fejl");

20

21 private String description;

22

23 Type(String description) {

24 this.description = description;

25 }

26

27 public String getDescription() {

28 return description;

29 }

30

31 public static List<String> getDescriptions() {

32 List<String> descriptions = new ArrayList<String>();

33 descriptions.add(ERROR.description);

34 descriptions.add(REQUEST.description);

35 return descriptions;

36 }

37

38 public static Type getType(String description) {

39 if (REQUEST.description.equals(description)) return REQUEST;

40 return ERROR;

41 }

42 }

43

44 /\*\*

45 \* Case status: NEW, CONSIDER, APPROVED, REJECTED and DONE.

46 \*/

47 public enum CaseStatus {

48 NEW("Ny"),

49 CONSIDERING("Behandles"),

50 APPROVED("Godkendt"),

51 REJECTED("Afvist"),

52 DONE("Afsluttet");

53

54 private String description;

55

56 CaseStatus(String description) {

57 this.description = description;

58 }

59

60 public String getDescription() {

61 return description;

62 }

63

64 public static List<String> getDescriptions() {

65 List<String> descriptions = new ArrayList<String>();

66 descriptions.add(NEW.description);

67 descriptions.add(CONSIDERING.description);

68 descriptions.add(APPROVED.description);

69 descriptions.add(REJECTED.description);

70 descriptions.add(DONE.description);

71 return descriptions;

72 }

73

74 public static CaseStatus getCaseStatus(String description) {

75 if (NEW.description.equals(description)) return NEW;

76 if (CONSIDERING.description.equals(description)) return CONSIDERING;

77 if (APPROVED.description.equals(description)) return APPROVED;

78 if (DONE.description.equals(description)) return DONE;

79 return REJECTED;

80 }

81 }

82

83 /\*\*

84 \* Developer status: NOTSTARTED, STARTED, READYTOTEST, TESTED and

85 \* INPRODUCTION.

86 \*/

87 public enum DevStatus {

88 NOTSTARTED("Ej begyndt"),

89 STARTED("I gang"),

90 READYTOTEST("Klar til test"),

91 TESTED("Testet"),

92 INPRODUCTION("I prod.");

93

94 private String description;

95

96 DevStatus(String description) {

97 this.description = description;

98 }

99

100 public String getDescription() {

101 return description;

102 }

103

104 public static List<String> getDescriptions() {

105 List<String> descriptions = new ArrayList<String>();

106 descriptions.add(NOTSTARTED.description);

107 descriptions.add(STARTED.description);

108 descriptions.add(READYTOTEST.description);

109 descriptions.add(TESTED.description);

110 descriptions.add(INPRODUCTION.description);

111 return descriptions;

112 }

113

114 public static DevStatus getDevStatus(String description) {

115 if (NOTSTARTED.description.equals(description)) return NOTSTARTED;

116 if (STARTED.description.equals(description)) return STARTED;

117 if (READYTOTEST.description.equals(description)) return READYTOTEST;

118 if (TESTED.description.equals(description)) return TESTED;

119 return INPRODUCTION;

120 }

121 }

122

123 /\*\*

124 \* BaRI user role: ADMIN, DEVELOPER, NORMAL.

125 \*/

126 public enum UserRole {

127 ADMIN("Administrator"),

128 DEVELOPER("Udvikler"),

129 NORMAL("Alm. bruger");

130

131 private String name;

132

133 UserRole(String name) {

134 this.name = name;

135 }

136

137 public String getName() {

138 return name;

139 }

140

141 public static List<String> getNames() {

142 List<String> names = new ArrayList<String>();

143 names.add(ADMIN.name);

144 names.add(DEVELOPER.name);

145 names.add(NORMAL.name);

146 return names;

147 }

148

149 public static UserRole getName(String name) {

150 if (ADMIN.name.equals(name)) {

151 return ADMIN;

152 }

153 else if (DEVELOPER.name.equals(name)) {

154 return DEVELOPER;

155 }

156 return NORMAL;

157 }

158 }

159 }

160

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\domain\DiscussionMessage.java |

1 package dk.jsh.itdiplom.dbsw.bari.domain;

2

3 import java.io.Serializable;

4 import java.util.Date;

5 import javax.persistence.\*;

6

7 /\*\*

8 \* Discussion message entity class.

9 \*

10 \* @author Jan S. Hansen

11 \*/

12 @Entity

13 public class DiscussionMessage implements Serializable {

14 private static final long serialVersionUID = 1L;

15

16 @Id

17 @GeneratedValue(strategy = GenerationType.IDENTITY)

18 protected Long id;

19 @Version

20 @Column(nullable = false)

21 protected Integer version;

22 @ManyToOne(optional=false)

23 @org.hibernate.annotations.ForeignKey(name="fk\_from\_discussionmessage\_to\_baricase")

24 protected BariCase bariCase;

25 @Temporal(javax.persistence.TemporalType.TIMESTAMP)

26 @Column(nullable = false)

27 protected Date created;

28 @ManyToOne(optional=false)

29 @org.hibernate.annotations.ForeignKey(name="fk\_from\_discussiommessage\_to\_bariuser")

30 protected BariUser bariUser;

31 @Column(length=400, nullable = false)

32 protected String message;

33

34 public DiscussionMessage() {

35 }

36

37 public DiscussionMessage(BariCase bariCase, Date created, BariUser bariUser,

38 String message) {

39 this.bariCase = bariCase;

40 this.created = created;

41 this.bariUser = bariUser;

42 this.message = message;

43 }

44

45 public BariUser getBariUser() {

46 return bariUser;

47 }

48

49 public void setBariUser(BariUser bariUser) {

50 this.bariUser = bariUser;

51 }

52

53 public Long getId() {

54 return id;

55 }

56

57 public void setId(Long id) {

58 this.id = id;

59 }

60

61 public Integer getVersion() {

62 return version;

63 }

64

65 public void setVersion(Integer version) {

66 this.version = version;

67 }

68

69 public BariCase getBariCase() {

70 return bariCase;

71 }

72

73 public void setBariCase(BariCase bariCase) {

74 this.bariCase = bariCase;

75 }

76

77 public Date getCreated() {

78 return created;

79 }

80

81 public void setCreated(Date created) {

82 this.created = created;

83 }

84

85 public String getMessage() {

86 return message;

87 }

88

89 public void setMessage(String message) {

90 this.message = message;

91 }

92 }

93

94

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\domain\Product.java |

1 package dk.jsh.itdiplom.dbsw.bari.domain;

2

3 import java.io.Serializable;

4 import javax.persistence.\*;

5

6 /\*\*

7 \* Product entity class.

8 \*

9 \* @author Jan S. Hansen

10 \*/

11 @Entity

12 public class Product implements Serializable {

13 private static final long serialVersionUID = 1L;

14

15 @Id

16 @GeneratedValue(strategy = GenerationType.IDENTITY)

17 protected Long id;

18 @Version

19 @Column(nullable = false)

20 protected Integer version;

21 @Column(length=50, nullable = false, unique=true)

22 protected String name;

23

24 public Product() {

25 }

26

27 public Product(String name) {

28 this.name = name;

29 }

30

31 public Long getId() {

32 return id;

33 }

34

35 public void setId(Long id) {

36 this.id = id;

37 }

38

39 public Integer getVersion() {

40 return version;

41 }

42

43 public String getName() {

44 return name;

45 }

46

47 public void setName(String name) {

48 this.name = name;

49 }

50 }

51

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\domain\UserGroup.java |

1 package dk.jsh.itdiplom.dbsw.bari.domain;

2

3 import java.io.Serializable;

4 import javax.persistence.\*;

5

6 /\*\*

7 \* UserGroup entity class.

8 \*

9 \* @author Jan S. Hansen

10 \*/

11 @Entity

12 @Table(name="UserGroup",

13 uniqueConstraints = {

14 @UniqueConstraint(columnNames={"bariUser\_id", "product\_id"})

15 })

16 public class UserGroup implements Serializable {

17 private static final long serialVersionUID = 1L;

18

19 @Id

20 @GeneratedValue(strategy = GenerationType.IDENTITY)

21 protected Long id;

22 @Version

23 @Column(nullable = false)

24 protected Integer version;

25 @ManyToOne(optional=false)

26 @org.hibernate.annotations.ForeignKey(name="fk\_from\_usergroup\_to\_bariuser")

27 protected BariUser bariUser;

28 @ManyToOne(optional=false)

29 @org.hibernate.annotations.ForeignKey(name="fk\_from\_usergroup\_to\_product")

30 protected Product product;

31

32 public UserGroup() {

33 }

34

35 public UserGroup(BariUser bariUser, Product product) {

36 this.bariUser = bariUser;

37 this.product = product;

38 }

39

40 public Long getId() {

41 return id;

42 }

43

44 public void setId(Long id) {

45 this.id = id;

46 }

47

48 public Integer getVersion() {

49 return version;

50 }

51

52 public BariUser getBariUser() {

53 return bariUser;

54 }

55

56 public void setBariUser(BariUser bariUser) {

57 this.bariUser = bariUser;

58 }

59

60 public Product getProduct() {

61 return product;

62 }

63

64 public void setProduct(Product product) {

65 this.product = product;

66 }

67 }

68

### dk.jsh.itdiplom.dbsw.bari.business

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\business\BariCaseBusiness.java |

1 package dk.jsh.itdiplom.dbsw.bari.business;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.BariCase;

4 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.CaseStatus;

5 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.Type;

6 import dk.jsh.itdiplom.dbsw.bari.domain.Product;

7 import dk.jsh.itdiplom.dbsw.bari.util.HibernateUtil;

8 import java.util.ArrayList;

9 import java.util.Date;

10 import java.util.List;

11 import org.hibernate.Query;

12 import org.hibernate.Session;

13 import org.hibernate.Transaction;

14

15 /\*\*

16 \* Business metods for BariCase.

17 \*

18 \* @author Jan S. Hansen

19 \*/

20 public class BariCaseBusiness {

21 /\*\*

22 \* Create new BariCase.

23 \*

24 \* @param bariCase BariCase

25 \*/

26 public static void saveNew(BariCase bariCase) {

27 Session session = HibernateUtil.getSessionFactory().openSession();

28 Transaction tx = session.beginTransaction();

29 session.save(bariCase);

30 tx.commit();

31 session.close();

32 }

33

34 /\*\*

35 \* Get all BariCase Eroor or Request objects for a product.

36 \*

37 \* @param product used in search.

38 \* @param type used in search.

39 \* @param caseStatus used in search, can be null.

40 \* @return a List of BariCase objects.

41 \*/

42 public static List<BariCase> getAllBariCases(Product product,

43 Type type, CaseStatus caseStatus) {

44 List<BariCase> bariCases = new ArrayList<BariCase>();

45 Session session = HibernateUtil.getSessionFactory().openSession();

46 String hql = "select bariCase from "

47 + "dk.jsh.itdiplom.dbsw.bari.domain.BariCase bariCase "

48 + "where bariCase.type = :type and "

49 + "product.id = :productid ";

50 if (caseStatus != null) {

51 hql += "and bariCase.caseStatus = :caseStatus ";

52 }

53 hql += "order by bariCase.created desc";

54 Query query = session.createQuery(hql);

55 query.setString("type", type.toString());

56 query.setString("productid", product.getId().toString());

57 if (caseStatus != null) {

58 query.setString("caseStatus", caseStatus.toString());

59 }

60 bariCases = query.list();

61 session.close();

62 return bariCases;

63 }

64

65 /\*\*

66 \* Update a BariCase.

67 \*

68 \* @param bariCase BariCase

69 \*/

70 public static void update(BariCase bariCase) {

71 Session session = HibernateUtil.getSessionFactory().openSession();

72 try {

73 Transaction tx = session.beginTransaction();

74 if (bariCase.getCaseStatus().equals(CaseStatus.DONE)) {

75 bariCase.setFinished(new Date());

76 }

77 session.update(bariCase);

78 tx.commit();

79 }

80 finally {

81 session.close();

82 }

83 }

84

85 /\*\*

86 \* Delete a bariCase and all discussion messages.

87 \*

88 \* @param bariCase BariCase

89 \*/

90 public static void delete(BariCase bariCase) {

91 Session session = HibernateUtil.getSessionFactory().openSession();

92 Transaction tx = session.beginTransaction();

93 String sql = "delete from DiscussionMessage where "

94 + "bariCase\_id = :id";

95 Query query = session.createSQLQuery(sql);

96 query.setString("id", bariCase.getId().toString());

97 query.executeUpdate();

98 session.delete(bariCase);

99 tx.commit();

100 session.close();

101 }

102 }

103

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\business\BariUserBusiness.java |

1 package dk.jsh.itdiplom.dbsw.bari.business;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

4 import dk.jsh.itdiplom.dbsw.bari.util.HibernateUtil;

5 import java.util.List;

6 import org.hibernate.Query;

7 import org.hibernate.Session;

8

9 /\*\*

10 \* Business metods for BariUser.

11 \*

12 \* @author Jan S. Hansen

13 \*/

14 public class BariUserBusiness {

15

16 /\*\*

17 \* Gets a bariUser from login and password.

18 \*

19 \* @param login bari user login

20 \* @param password password

21 \* @return a BariUser or null if login or password is wrong.

22 \*/

23 public static BariUser isValidUser(String login, String password) {

24 BariUser bariUser = null;

25 Session session = HibernateUtil.getSessionFactory().openSession();

26 String hql = "select bariUser from "

27 + "dk.jsh.itdiplom.dbsw.bari.domain.BariUser bariUser "

28 + "where bariUser.login = :login "

29 + "and bariUser.password = :password";

30 Query query = session.createQuery(hql);

31 query.setString("login", login);

32 query.setString("password", login);

33 List<BariUser> bariUsers = query.list();

34 if (bariUsers.size() == 1) {

35 bariUser = bariUsers.get(0);

36 }

37 else if (bariUsers.size() > 1) {

38 throw new RuntimeException("More then one user with login " +

39 login);

40 }

41 session.close();

42 return bariUser;

43 }

44 }

45

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\business\DiscussionMessageBusiness.java |

1 package dk.jsh.itdiplom.dbsw.bari.business;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.BariCase;

4 import dk.jsh.itdiplom.dbsw.bari.domain.DiscussionMessage;

5 import dk.jsh.itdiplom.dbsw.bari.util.HibernateUtil;

6 import java.util.ArrayList;

7 import java.util.List;

8 import org.hibernate.Query;

9 import org.hibernate.Session;

10 import org.hibernate.Transaction;

11

12 /\*\*

13 \* Business metods for DescussionMessage.

14 \*

15 \* @author Jan S. Hansen

16 \*/

17 public class DiscussionMessageBusiness {

18 /\*\*

19 \* Create new DiscussionMessage.

20 \*

21 \* @param discussionMessage DiscussionMessage

22 \*/

23 public static void saveNew(DiscussionMessage discussionMessage) {

24 Session session = HibernateUtil.getSessionFactory().openSession();

25 Transaction tx = session.beginTransaction();

26 session.save(discussionMessage);

27 tx.commit();

28 session.close();

29 }

30

31 /\*\*

32 \* Get all DiscussionMessages for a specific BariCase.

33 \*

34 \* @param bariCase BariCase.

35 \* @return a List of DiscussionMessage objects.

36 \*/

37 public static List<DiscussionMessage> getAllDiscussionMessages(

38 BariCase bariCase) {

39 List<DiscussionMessage> discussionMessages =

40 new ArrayList<DiscussionMessage>();

41 Session session = HibernateUtil.getSessionFactory().openSession();

42 String hql = "select discussionMessage from "

43 + "dk.jsh.itdiplom.dbsw.bari.domain.DiscussionMessage discussionMessage "

44 + "where bariCase.id = :id "

45 + "order by discussionMessage.created";

46 Query query = session.createQuery(hql);

47 query.setString("id", bariCase.getId().toString());

48 discussionMessages = query.list();

49 session.close();

50 return discussionMessages;

51 }

52 }

53

54

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\business\UserGroupBusiness.java |

1 package dk.jsh.itdiplom.dbsw.bari.business;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

4 import dk.jsh.itdiplom.dbsw.bari.domain.Product;

5 import dk.jsh.itdiplom.dbsw.bari.util.HibernateUtil;

6 import java.util.ArrayList;

7 import java.util.List;

8 import org.hibernate.Query;

9 import org.hibernate.Session;

10

11 /\*\*

12 \* Business metods for UserGroup.

13 \*

14 \* @author Jan S. Hansen

15 \*/

16 public class UserGroupBusiness {

17

18 /\*\*

19 \* Get all Products a given bariUser can access.

20 \*

21 \* @param bariCase BariCase.

22 \* @return a List of DiscussionMessage objects.

23 \*/

24 public static List<Product> getAllDiscussionMessages(

25 BariUser bariUser) {

26 List<Product> products =

27 new ArrayList<Product>();

28 Session session = HibernateUtil.getSessionFactory().openSession();

29 String hql = "select product from "

30 + "dk.jsh.itdiplom.dbsw.bari.domain.Product product, "

31 + "dk.jsh.itdiplom.dbsw.bari.domain.UserGroup userGroup "

32 + "where userGroup.bariUser.id = :userid and "

33 + "product.id = userGroup.product.id "

34 + "order by product.name";

35 Query query = session.createQuery(hql);

36 query.setString("userid", bariUser.getId().toString());

37 products = query.list();

38 session.close();

39 return products;

40 }

41 }

42

### dk.jsh.itdiplom.dbsw.wicket

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\About.html |

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

2 <html>

3 <head>

4 <title></title>

5 </head>

6 <body>

7 <wicket:extend>

8 <fieldset>

9 <legend>Om BaRI</legend>

10 <p>BaRI st&aring;r for <b>B</b>ugs <b>a</b>nd <b>R</b>equest

11 <b>I</b>nterceptor</p>

12 <p>Programmet er udviklet i forbindelse med faget Databasesystemer og web,

13 som er et fag under IT-Diplomuddannelsen.</p>

14 <p>Programmet er udviklet af Jan Schr&oslash;der Hansen,

15 efter&aring;ret 2010.</p>

16 <p>e-mail <a href="mailto:jan.sch.hansen@gmail.com">

17 jan.sch.hansen@gmail.com</a></p>

18 </fieldset>

19 <fieldset>

20 <legend>M&aring;ling af kodekvalitet -

21 "What the fuck" per minute</legend>

22 <img wicket:id="wtfm"/>

23 </fieldset>

24 </wicket:extend>

25 </body>

26 </html>

27

28

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\About.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2

3 import org.apache.wicket.markup.html.image.Image;

4

5 /\*\*

6 \* About page.

7 \*

8 \* @author Jan S. Hansen

9 \*/

10 public class About extends BasePage {

11

12 public About() {

13 add(new Image("wtfm", "wtfm.jpg"));

14 }

15 }

16

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Application.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2

3 import org.apache.wicket.Request;

4 import org.apache.wicket.Response;

5 import org.apache.wicket.Session;

6 import org.apache.wicket.protocol.http.WebApplication;

7

8 /\*\*

9 \* Wicket application.

10 \*

11 \* @author Jan S. Hansen

12 \*/

13 public class Application extends WebApplication {

14

15 /\*\*

16 \* Constructor.

17 \*/

18 public Application() {

19 }

20

21 /\*\*

22 \* Returns home page for the application.

23 \*/

24 @Override

25 public Class getHomePage() {

26 return Login.class;

27 }

28

29 @Override

30 public Session newSession(Request request, Response response) {

31 return new BariSession(request);

32 }

33 }

34

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\BariSession.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

4 import org.apache.wicket.Request;

5 import org.apache.wicket.Session;

6 import org.apache.wicket.protocol.http.WebSession;

7

8 /\*\*

9 \* Wicket/BaRI session.

10 \*

11 \* @author Jan S. Hansen

12 \*/

13 public class BariSession extends WebSession {

14 private BariUser bariUser;

15

16 public BariSession(Request request) {

17 super(request);

18 }

19

20 public static BariSession get() {

21 return (BariSession) Session.get();

22 }

23

24 public boolean isAuthenticated() {

25 return (bariUser != null);

26 }

27

28 public BariUser getBariUser() {

29 return bariUser;

30 }

31

32 public void setBariUser(BariUser bariUser) {

33 this.bariUser = bariUser;

34 }

35 }

36

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\BasePage.html |

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

2 <html>

3 <head>

4 <title>BaRI</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

6 <link wicket:id='stylesheet'/>

7 </head>

8 <body>

9 <div id="container">

10 <div id="header">

11 <h1>BaRI</h1>

12 </div>

13 <div id="contents">

14 <fieldset>

15 <legend>Bruger og rolle</legend>

16 <span wicket:id="userandrole">User and role goes here</span>

17 </fieldset>

18 <fieldset>

19 <legend>Menu</legend>

20 <a href="#" wicket:id="createnew">Opret</a>

21 <a href="#" wicket:id="overview">Oversigt</a>

22 <a href="#" wicket:id="logout">Log ud</a>

23 <a style="float:right" href="#" wicket:id="about">

24 Om BaRI</a>

25 </fieldset>

26 </div>

27 <div id="error">

28 <span wicket:id="error">Error message goes here</span>

29 </div>

30 <div id="main">

31 <wicket:child />

32 </div>

33 </div>

34 </body>

35 </html>

36

37

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\BasePage.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2

3 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

4 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.Type;

5 import java.text.SimpleDateFormat;

6 import org.apache.wicket.Page;

7 import org.apache.wicket.markup.html.WebPage;

8 import org.apache.wicket.markup.html.basic.Label;

9 import org.apache.wicket.markup.html.link.Link;

10 import org.apache.wicket.markup.html.resources.StyleSheetReference;

11 import org.apache.wicket.model.Model;

12 import org.apache.wicket.model.PropertyModel;

13

14 /\*\*

15 \* Abstract wicket base page. Handles common error message handling, stylecheet

16 \* and menu links.

17 \*

18 \* @author Jan S. Hansen

19 \*/

20 public abstract class BasePage extends WebPage {

21 final static SimpleDateFormat standardDateTimeFormat =

22 new SimpleDateFormat("dd/MM-yyyy HH:mm");

23 private String errorMessage = "";

24 private BariUser bariUser;

25

26 /\*\*

27 \* Constructor

28 \*/

29 public BasePage() {

30 BariSession bariSession = BariSession.get();

31 bariUser = bariSession.getBariUser();

32 if (bariUser == null) {

33 Page loginPage = new Login();

34 this.setResponsePage(loginPage);

35 }

36 add(new Label("userandrole", new Model(bariUser.getFullname() + " som "

37 + bariUser.getUserRole().getName())));

38

39 PropertyModel errorMessageModel =

40 new PropertyModel(this, "errorMessage");

41 add(new Label("error", errorMessageModel));

42 add(new StyleSheetReference("stylesheet", BasePage.class, "style.css"));

43

44 add(new Link("createnew") {

45 @Override

46 public void onClick() {

47 Page page = new CreateNew();

48 setResponsePage(page);

49 }

50 });

51

52 add(new Link("overview") {

53 @Override

54 public void onClick() {

55 Page page = new Overview(null, Type.ERROR, "Alle");

56 setResponsePage(page);

57 }

58 });

59

60 add(new Link("logout") {

61 @Override

62 public void onClick() {

63 BariSession bariSession = BariSession.get();

64 bariSession.setBariUser(null);

65 Page page = new Login();

66 setResponsePage(page);

67 }

68 });

69

70 add(new Link("about") {

71 @Override

72 public void onClick() {

73 Page page = new About();

74 setResponsePage(page);

75 }

76 });

77 }

78

79 /\*\*

80 \* Get BaRI user.

81 \*/

82 public BariUser getBariUser() {

83 return bariUser;

84 }

85

86 /\*\*

87 \* Set BaRI user.

88 \*/

89 public void setBariUser(BariUser bariUser) {

90 this.bariUser = bariUser;

91 }

92

93 /\*\*

94 \* Returns an error message

95 \*/

96 public String getErrorMessage() {

97 return errorMessage;

98 }

99

100 /\*\*

101 \* Set error message.

102 \*/

103 public void setErrorMessage(String errorMessage) {

104 this.errorMessage = errorMessage;

105 }

106 }

107

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\CreateNew.html |

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

4 <html xmlns:wicket="http://wicket.apache.org">

5 <head>

6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

7 <title></title>

8 <link rel="stylesheet" type="text/css" href="style.css"/>

9 </head>

10 <body>

11 <wicket:extend>

12 <fieldset>

13 <legend>Opret</legend>

14 <form action="#" wicket:id="form">

15 <table align="center" border="0">

16 <tbody>

17 <tr>

18 <td style="width:120px">Overskrift:</td>

19 <td>

20 <input wicket:id="title" type="text" name="" value="" size="50" />

21 </td>

22 </tr>

23 <tr>

24 <td>Product:</td>

25 <td><select wicket:id="product" name="">

26 <option>BaRI</option>

27 <option>xxx</option>

28 </select></td>

29 </tr>

30 <tr>

31 <td>Type:</td>

32 <td><select wicket:id="type" name="">

33 <option>Nyt ønske</option>

34 <option>Fejl</option>

35 </select></td>

36 </tr>

37 <tr>

38 <td>Beskrivelse:</td>

39 <td>&nbsp;</td>

40 </tr>

41 <tr>

42 <td colspan="2">

43 <textarea wicket:id="description" name="" rows="5" cols="80" >

44 </textarea>

45 </td>

46 </tr>

47 <tr>

48 <td colspan="2">

49 <div style="float:right">

50 <input wicket:id="cancel" type="button" value="Fortryd" />

51 <input wicket:id="save" type="submit" value="Gem" />

52 </div>

53 </td>

54 </tr>

55 </tbody>

56 </table>

57 </form>

58 </fieldset>

59 </wicket:extend>

60 </body>

61 </html>

62

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\CreateNew.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2 import dk.jsh.itdiplom.dbsw.bari.business.BariCaseBusiness;

3 import dk.jsh.itdiplom.dbsw.bari.business.UserGroupBusiness;

4 import dk.jsh.itdiplom.dbsw.bari.domain.BariCase;

5 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

6 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.CaseStatus;

7 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.DevStatus;

8 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.Type;

9 import dk.jsh.itdiplom.dbsw.bari.domain.Product;

10 import java.util.ArrayList;

11 import java.util.Date;

12 import java.util.List;

13 import org.apache.wicket.AttributeModifier;

14 import org.apache.wicket.markup.html.form.Button;

15 import org.apache.wicket.markup.html.form.ChoiceRenderer;

16 import org.apache.wicket.markup.html.form.DropDownChoice;

17 import org.apache.wicket.markup.html.form.Form;

18 import org.apache.wicket.markup.html.form.TextArea;

19 import org.apache.wicket.markup.html.form.TextField;

20 import org.apache.wicket.markup.html.link.Link;

21 import org.apache.wicket.model.Model;

22 import org.apache.wicket.validation.validator.StringValidator;

23

24 /\*\*

25 \* Create new BariCase page.

26 \*

27 \* @author Jan S. Hansen

28 \*/

29 public final class CreateNew extends BasePage {

30 private TextField<String> title;

31 private DropDownChoice<Product> products;

32 private Product selectedProduct;

33 private DropDownChoice<String> type;

34 private TextArea<String> description;

35

36 /\*\*

37 \* Constructor.

38 \*/

39 public CreateNew() {

40 BariUser bariUser = BariSession.get().getBariUser();

41 List<Product> listOfProducts =

42 UserGroupBusiness.getAllDiscussionMessages(bariUser);

43 selectedProduct = listOfProducts.get(0);

44

45 //Add a form as an inner class.

46 Form form = new Form("form") {

47 //Handles required fields error.

48 @Override

49 protected void onError() {

50 List<String> emptyFields = new ArrayList<String>();

51 if (!title.checkRequired()) {

52 emptyFields.add("Overskrift");

53 title.add(new AttributeModifier("style", true,

54 new Model("border-color:red;")));

55 }

56 else {

57 title.add(new AttributeModifier("style", true,

58 new Model("border-color:default;")));

59 }

60 if (!products.checkRequired()) {

61 emptyFields.add("Produkt");

62 products.add(new AttributeModifier("style", true,

63 new Model("border-color:red;")));

64 }

65 else {

66 products.add(new AttributeModifier("style", true,

67 new Model("border-color:default;")));

68 }

69 if (!type.checkRequired()) {

70 emptyFields.add("Type");

71 type.add(new AttributeModifier("style", true,

72 new Model("border-color:red;")));

73 }

74 else {

75 type.add(new AttributeModifier("style", true,

76 new Model("border-color:default;")));

77 }

78 if (!description.checkRequired()) {

79 emptyFields.add("Beskrivelse");

80 description.add(new AttributeModifier("style", true,

81 new Model("border-color:red;")));

82 }

83 else {

84 description.add(new AttributeModifier("style", true,

85 new Model("border-color:defalut;")));

86 }

87 StringBuilder errorMessage = new StringBuilder();

88 if (emptyFields.size() > 0) {

89 if (emptyFields.size() == 1) {

90 errorMessage.append("Feltet ");

91 errorMessage.append("'").append(emptyFields.get(0))

92 .append("'");

93 }

94 else {

95 errorMessage.append("Felterne ");

96 int fieldCounter = 1;

97 for (String field : emptyFields) {

98 errorMessage.append("'").append(field).append("'");

99 if (fieldCounter < emptyFields.size() -1) {

100 errorMessage.append(", ");

101 }

102 if (fieldCounter == emptyFields.size() -1) {

103 errorMessage.append(" og ");

104 }

105 fieldCounter++;

106 }

107 }

108 errorMessage.append(" skal udfyldes.");

109 }

110 if (!description.isValid()) {

111 if (errorMessage.length() > 0) {

112 errorMessage.append(" ");

113 }

114 errorMessage.append("Beskrivelses feltet kan max. " +

115 "være 255 tegn langt.");

116 description.add(new AttributeModifier("style", true,

117 new Model("border-color:red;")));

118 }

119 else {

120 description.add(new AttributeModifier("style", true,

121 new Model("border-color:defalut;")));

122 }

123 setErrorMessage(errorMessage.toString());

124 }

125 };

126 add(form);

127

128 //Add fields to the form.

129 title = new TextField("title", new Model(""));

130 title.setRequired(true);

131 form.add(title);

132 products = new DropDownChoice("product", new Model(selectedProduct),

133 listOfProducts, new ChoiceRenderer("name", "id"));

134 products.setRequired(true);

135 form.add(products);

136 type = new DropDownChoice("type",

137 new Model(Type.ERROR.getDescription()), Type.getDescriptions());

138 type.setRequired(true);

139 form.add(type);

140 description = new TextArea("description", new Model(""));

141 description.setRequired(true);

142 description.add(StringValidator.maximumLength(255));

143 form.add(description);

144

145 //Add buttons to the form.

146 form.add(new Link("cancel") {

147 @Override

148 public void onClick() {

149 setResponsePage(Overview.class);

150 }

151 });

152 form.add(new Button("save") {

153 @Override

154 public void onSubmit() {

155 BariUser bariUser = BariSession.get().getBariUser();

156 BariCase bariCase = new BariCase(

157 title.getModelObject(),

158 Type.getType(type.getModelObject()),

159 bariUser, products.getModelObject(),

160 new Date(), null, CaseStatus.NEW,

161 DevStatus.NOTSTARTED,

162 description.getModelObject(),

163 null);

164 BariCaseBusiness.saveNew(bariCase);

165 setResponsePage(Overview.class);

166 }

167 });

168 }

169 }

170

171

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Discussion.html |

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

4 <html xmlns:wicket="http://wicket.apache.org">

5 <head>

6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

7 <title>Discussion</title>

8 <link rel="stylesheet" type="text/css" href="style.css"/>

9 </head>

10 <body>

11 <wicket:extend>

12 <fieldset>

13 <legend>Diskussion</legend>

14 <table align="center" border="0">

15 <tbody>

16 <tr>

17 <td colspan="2">

18 <b><span wicket:id="product">BaRI</span>&nbsp;-&nbsp;

19 <span wicket:id="type">Fejl</span>:&nbsp;

20 <span wicket:id="title">Overskrift</span>

21 </b>

22 </td>

23 </tr>

24 <tr>

25 <td colspan="2">

26 <textarea wicket:id="log" name="" rows="15" cols="90"

27 readonly="readonly">

28 </textarea>

29 </td>

30 </tr>

31

32 </tbody>

33 </table>

34 <form action="#" wicket:id="form">

35 <table align="center" border="0">

36 <tbody>

37 <tr>

38 <td colspan="2">

39 <b>Nyt indlæg:</b>

40 </td>

41 </tr>

42 <tr>

43 <td colspan="2">Indlæg:</td>

44 </tr>

45 <tr>

46 <td colspan="2">

47 <textarea wicket:id="message" name=""

48 rows="3" cols="80">

49 </textarea>

50 </td>

51 </tr>

52 <tr>

53 <td colspan="2" >

54 <a href="#" wicket:id="goBack">Tilbage</a>

55 <div style="float:right">

56 <input wicket:id="save" type="submit"

57 value="Gem nyt indlæg" />

58 </div>

59 </td>

60 </tr>

61 </tbody>

62 </table>

63 </form>

64 </fieldset>

65 </wicket:extend>

66 </body>

67 </html>

68

69

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Discussion.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2 import dk.jsh.itdiplom.dbsw.bari.business.DiscussionMessageBusiness;

3 import dk.jsh.itdiplom.dbsw.bari.domain.BariCase;

4 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

5 import dk.jsh.itdiplom.dbsw.bari.domain.DiscussionMessage;

6 import java.util.ArrayList;

7 import java.util.Date;

8 import java.util.List;

9 import org.apache.wicket.AttributeModifier;

10 import org.apache.wicket.Page;

11 import org.apache.wicket.markup.html.basic.Label;

12 import org.apache.wicket.markup.html.form.Button;

13 import org.apache.wicket.markup.html.form.Form;

14 import org.apache.wicket.markup.html.form.TextArea;

15 import org.apache.wicket.markup.html.form.TextField;

16 import org.apache.wicket.markup.html.link.Link;

17 import org.apache.wicket.model.Model;

18

19 /\*\*

20 \* Discussion page.

21 \*

22 \* @author Jan S. Hansen

23 \*/

24 public final class Discussion extends BasePage {

25 private TextField<String> user;

26 private TextArea<String> message;

27

28 /\*\*

29 \* Constructor.

30 \*

31 \* @param bariCase A BaRICase.

32 \*/

33 public Discussion(final BariCase bariCase) {

34 add(new Label("product", new Model(bariCase.getProduct().getName())));

35 add(new Label("type", new Model(bariCase.getType().getDescription())));

36 add(new Label("title", new Model(bariCase.getTitle())));

37

38 //Get all discussion messages and build a discussion log.

39 List<DiscussionMessage> discussionMessages =

40 DiscussionMessageBusiness.getAllDiscussionMessages(bariCase);

41 if (discussionMessages.size() > 0) {

42 StringBuilder log = new StringBuilder();

43 for (DiscussionMessage discussionMessage : discussionMessages) {

44 log.append(standardDateTimeFormat.format(

45 discussionMessage.getCreated()));

46 log.append(" af ");

47 log.append(discussionMessage.getBariUser().getFullname());

48 log.append(":\n");

49 log.append(discussionMessage.getMessage());

50 log.append("\n\n");

51 }

52 add(new TextArea("log", new Model(log.toString())));

53 }

54 else {

55 add(new TextArea("log", new Model("Der er ingen indlæg.")));

56 }

57

58 //Create and add a form

59 Form form = new Form("form") {

60 //Handles required fields error.

61 @Override

62 protected void onError() {

63 List<String> emptyFields = new ArrayList<String>();

64 if (!message.checkRequired()) {

65 emptyFields.add("Indlæg");

66 message.add(new AttributeModifier("style", true,

67 new Model("border-color:red;")));

68 }

69 else {

70 message.add(new AttributeModifier("style", true,

71 new Model("border-color:default;")));

72 }

73 StringBuilder errorMessage = new StringBuilder();

74 if (emptyFields.size() == 1) {

75 errorMessage.append("Feltet ");

76 errorMessage.append("'").append(emptyFields.get(0))

77 .append("'");

78 }

79 else {

80 errorMessage.append("Felterne ");

81 int fieldCounter = 1;

82 for (String field : emptyFields) {

83 errorMessage.append("'").append(field).append("'");

84 if (fieldCounter < emptyFields.size() -1) {

85 errorMessage.append(", ");

86 }

87 if (fieldCounter == emptyFields.size() -1) {

88 errorMessage.append(" og ");

89 }

90 fieldCounter++;

91 }

92 }

93 errorMessage.append(" skal udfyldes.");

94 setErrorMessage(errorMessage.toString());

95 }

96 };

97 add(form);

98

99 //Add fields to the form.

100 message = new TextArea("message", new Model(""));

101 message.setRequired(true);

102 form.add(message);

103

104 //Add links and buttons to the form.

105 form.add(new Link("goBack") {

106 @Override

107 public void onClick() {

108 Page page = new Update(bariCase);

109 setResponsePage(page);

110 }

111 });

112

113 form.add(new Button("save") {

114 @Override

115 public void onSubmit() {

116 BariUser bariUser = BariSession.get().getBariUser();

117 DiscussionMessage discussionMessage = new DiscussionMessage(

118 bariCase, new Date(), bariUser,

119 message.getModelObject());

120 DiscussionMessageBusiness.saveNew(discussionMessage);

121 Page page = new Discussion(bariCase);

122 setResponsePage(page);

123 }

124 });

125 }

126 }

127

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Login.html |

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

2 <html>

3 <head>

4 <title>BaRI Login</title>

5 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

6 <link wicket:id='stylesheet'/>

7 </head>

8 <body>

9 <div id="container">

10 <div id="header">

11 <h1>BaRI</h1>

12 </div>

13 <div id="contents">

14 <fieldset>

15 <legend>Login</legend>

16 <form action="#" wicket:id="form">

17 <table align="center" border="0">

18 <tbody>

19 <tr>

20 <td style="width:120px">Login:</td>

21 <td>

22 <input wicket:id="userlogin" type="text"

23 name="" value="" size="20" />

24 </td>

25 </tr>

26 <tr>

27 <td style="width:120px">Password:</td>

28 <td>

29 <input wicket:id="password" type="password"

30 name="" value="" size="20" />

31 </td>

32 </tr>

33 <tr>

34 <td colspan="2">

35 <div style="float:right">

36 <input wicket:id="barilogin" type="submit"

37 value="Login" />

38 </div>

39 </td>

40 </tr>

41 </tbody>

42 </table>

43 </form>

44 </fieldset>

45 </div>

46 <div id="error">

47 <span wicket:id="error">Error message goes here</span>

48 </div>

49 </div>

50 </body>

51 </html>

52

53

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Login.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2

3 import dk.jsh.itdiplom.dbsw.bari.business.BariUserBusiness;

4 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

5 import java.util.ArrayList;

6 import java.util.List;

7 import org.apache.wicket.AttributeModifier;

8 import org.apache.wicket.markup.html.WebPage;

9 import org.apache.wicket.markup.html.basic.Label;

10 import org.apache.wicket.markup.html.form.Button;

11 import org.apache.wicket.markup.html.form.Form;

12 import org.apache.wicket.markup.html.form.PasswordTextField;

13 import org.apache.wicket.markup.html.form.TextField;

14 import org.apache.wicket.markup.html.resources.StyleSheetReference;

15 import org.apache.wicket.model.Model;

16 import org.apache.wicket.model.PropertyModel;

17

18 /\*\*

19 \* BaRI login page.

20 \*

21 \* @author Jan S. Hansen

22 \*/

23 public class Login extends WebPage {

24 private String errorMessage = "";

25 private TextField<String> userLogin;

26 private TextField<String> password;

27

28 /\*\*

29 \* Constructor.

30 \*/

31 public Login() {

32 add(new StyleSheetReference("stylesheet", BasePage.class, "style.css"));

33 PropertyModel errorMessageModel =

34 new PropertyModel(this, "errorMessage");

35 add(new Label("error", errorMessageModel));

36 //Add a form as an inner class.

37 Form form = new Form("form") {

38 //Handles required fields error.

39 @Override

40 protected void onError() {

41 List<String> emptyFields = new ArrayList<String>();

42 if (!userLogin.checkRequired()) {

43 emptyFields.add("Login");

44 userLogin.add(new AttributeModifier("style", true,

45 new Model("border-color:red;")));

46 }

47 else {

48 userLogin.add(new AttributeModifier("style", true,

49 new Model("border-color:default;")));

50 }

51 if (!password.checkRequired()) {

52 emptyFields.add("Password");

53 password.add(new AttributeModifier("style", true,

54 new Model("border-color:red;")));

55 }

56 else {

57 password.add(new AttributeModifier("style", true,

58 new Model("border-color:default;")));

59 }

60 StringBuilder errorMessage = new StringBuilder();

61 if (emptyFields.size() > 0) {

62 if (emptyFields.size() == 1) {

63 errorMessage.append("Feltet ");

64 errorMessage.append("'").append(emptyFields.get(0))

65 .append("'");

66 }

67 else {

68 errorMessage.append("Felterne ");

69 int fieldCounter = 1;

70 for (String field : emptyFields) {

71 errorMessage.append("'").append(field).append("'");

72 if (fieldCounter < emptyFields.size() -1) {

73 errorMessage.append(", ");

74 }

75 if (fieldCounter == emptyFields.size() -1) {

76 errorMessage.append(" og ");

77 }

78 fieldCounter++;

79 }

80 }

81 errorMessage.append(" skal udfyldes.");

82 }

83 setErrorMessage(errorMessage.toString());

84 }

85 };

86 add(form);

87

88 //Add fields to the form.

89 userLogin = new TextField("userlogin", new Model(""));

90 userLogin.setRequired(true);

91 form.add(userLogin);

92

93 password = new PasswordTextField("password", new Model(""));

94 password.setRequired(true);

95 form.add(password);

96

97

98 //Add button to the form.

99 form.add(new Button("barilogin") {

100 @Override

101 public void onSubmit() {

102 BariUser bariUser =

103 BariUserBusiness.isValidUser(userLogin.getModelObject(),

104 password.getModelObject());

105 if (bariUser != null) {

106 BariSession bariSession = BariSession.get();

107 bariSession.setBariUser(bariUser);

108 setResponsePage(Overview.class);

109 }

110 else {

111 setErrorMessage("Fejl i login eller password.");

112 }

113 }

114 });

115 }

116

117 /\*\*

118 \* Returns an error message

119 \*/

120 public String getErrorMessage() {

121 return errorMessage;

122 }

123

124 /\*\*

125 \* Set error message.

126 \*/

127 public void setErrorMessage(String errorMessage) {

128 this.errorMessage = errorMessage;

129 }

130 }

131

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Overview.html |

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

2 <html>

3 <head>

4 <title></title>

5 </head>

6 <body>

7 <wicket:extend>

8 <fieldset>

9 <legend>Oversigt</legend>

10 <form action="#" wicket:id="form">

11 Product:&nbsp;

12 <select wicket:id="product" name="">

13 <option>BaRI</option>

14 <option>xxx</option>

15 </select>

16 Type:&nbsp;

17 <select wicket:id="type" name="">

18 <option>Nyt ønske</option>

19 <option>Fejl</option>

20 </select>

21 Status:&nbsp;

22 <select wicket:id="status" name="">

23 <option>Alle</option>

24 <option>Ny</option>

25 <option>Under behandling</option>

26 <option>Godkendt</option>

27 <option>Afvist</option>

28 </select>

29 <input style="float:right" wicket:id="search" type="submit"

30 value="S&oslash;g" />

31 </form>

32 <table class="pageablelistview" border="0" cellspacing="0" width="100%">

33 <tr>

34 <th>Overskrift</th>

35 <th>Oprettet</th>

36 <th>Afsluttet</th>

37 <th>Status</th>

38 <th>Udvikling</th>

39 <th>&nbsp;</th>

40 </tr>

41 <tr wicket:id="pageable">

42 <td><span wicket:id="title">[title]</span></td>

43 <td><span wicket:id="created">[created]</span></td>

44 <td><span wicket:id="finished">[finished]</span></td>

45 <td><span wicket:id="casestatus">[casestatus]</span></td>

46 <td><span wicket:id="devstatus">[devstatus]</span></td>

47 <td><a href="#" wicket:id="action">Vis</a></td>

48 </tr>

49 </table>

50 <div style="float:right">

51 <span wicket:id="navigator">[dataview navigator]</span>

52 </div>

53 </fieldset>

54 </wicket:extend>

55 </body>

56 </html>

57

58

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Overview.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2

3 import dk.jsh.itdiplom.dbsw.bari.business.BariCaseBusiness;

4 import dk.jsh.itdiplom.dbsw.bari.business.UserGroupBusiness;

5 import dk.jsh.itdiplom.dbsw.bari.domain.BariCase;

6 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

7 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.CaseStatus;

8 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.Type;

9 import dk.jsh.itdiplom.dbsw.bari.domain.Product;

10 import java.util.LinkedList;

11 import java.util.List;

12 import org.apache.wicket.AttributeModifier;

13 import org.apache.wicket.Page;

14 import org.apache.wicket.markup.html.basic.Label;

15 import org.apache.wicket.markup.html.form.Button;

16 import org.apache.wicket.markup.html.form.ChoiceRenderer;

17 import org.apache.wicket.markup.html.form.DropDownChoice;

18 import org.apache.wicket.markup.html.form.Form;

19 import org.apache.wicket.markup.html.link.Link;

20 import org.apache.wicket.markup.html.list.ListItem;

21 import org.apache.wicket.markup.html.list.PageableListView;

22 import org.apache.wicket.markup.html.navigation.paging.PagingNavigator;

23 import org.apache.wicket.model.AbstractReadOnlyModel;

24 import org.apache.wicket.model.Model;

25

26 /\*\*

27 \* Overview page for all BariCases.

28 \*

29 \* @author Jan S. Hansen

30 \*/

31 public final class Overview extends BasePage {

32 private DropDownChoice<Product> products;

33 private Product selectedProduct;

34 private DropDownChoice<String> dropDownChoiceType;

35 private DropDownChoice<String> statusDownChoiceType;

36

37 /\*\*

38 \* Constructor.

39 \*/

40 public Overview() {

41 this(null, Type.ERROR, "Alle");

42 }

43

44 /\*\*

45 \* Constructor.

46 \*

47 \* @param product Product to use as default on overview page.

48 \* @param type Type used as default on overview page.

49 \* @param status Status uses as default on overview page.

50 \*/

51 public Overview(Product product, Type type, String status) {

52 BariUser bariUser = BariSession.get().getBariUser();

53 List<Product> listOfProducts =

54 UserGroupBusiness.getAllDiscussionMessages(bariUser);

55 if (product == null) {

56 selectedProduct = listOfProducts.get(0);

57 }

58 else {

59 selectedProduct = product;

60 }

61

62 //Add search form.

63 Form form = new Form("form");

64 add(form);

65 products = new DropDownChoice("product", new Model(selectedProduct),

66 listOfProducts, new ChoiceRenderer("name", "id"));

67 products.setRequired(true);

68 form.add(products);

69

70 dropDownChoiceType = new DropDownChoice("type",

71 new Model(type.getDescription()), Type.getDescriptions());

72 form.add(dropDownChoiceType);

73 LinkedList<String> statusList =

74 new LinkedList<String>(CaseStatus.getDescriptions());

75 statusList.addFirst("Alle");

76 statusDownChoiceType =

77 new DropDownChoice("status", new Model(status), statusList);

78 form.add(statusDownChoiceType);

79 form.add(new Button("search") {

80 @Override

81 public void onSubmit() {

82 Page page = new Overview(products.getModelObject(),

83 Type.getType(dropDownChoiceType.getModelObject()),

84 statusDownChoiceType.getModelObject());

85 setResponsePage(page);

86 }

87 });

88

89 //Add table with search results.

90 CaseStatus cs = null;

91 if (!"Alle".equals(statusDownChoiceType.getModelObject())) {

92 cs = CaseStatus.getCaseStatus(statusDownChoiceType.getModelObject());

93 }

94 List<BariCase> bariCases = BariCaseBusiness.getAllBariCases(selectedProduct,

95 type, cs);

96 PageableListView pageableListView =

97 new PageableListView("pageable", bariCases, 10) {

98 @Override

99 protected void populateItem(final ListItem item) {

100 final BariCase bariCase = (BariCase) item.getModelObject();

101 item.add(new Label("title", bariCase.getTitle()));

102 item.add(new Label("created",

103 standardDateTimeFormat.format(bariCase.getCreated())));

104 item.add(new Label("finished",

105 bariCase.getFinished() == null ? ""

106 :standardDateTimeFormat.format(bariCase.getFinished())));

107 item.add(new Label("casestatus",

108 bariCase.getCaseStatus().getDescription()));

109 item.add(new Label("devstatus",

110 bariCase.getDevStatus().getDescription()));

111 item.add(new Link("action") {

112 @Override

113 public void onClick() {

114 Page page = new Update(bariCase);

115 setResponsePage(page);

116 }

117 });

118 item.add(new AttributeModifier("class",

119 true, new AbstractReadOnlyModel<String>()

120 {

121 @Override

122 public String getObject()

123 {

124 return (item.getIndex() % 2 == 1) ? "even" : "odd";

125 }

126 }));

127 }

128 };

129 add(pageableListView);

130 add(new PagingNavigator("navigator", pageableListView));

131 }

132 }

133

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Update.html |

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

4 <html xmlns:wicket="http://wicket.apache.org">

5 <head>

6 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>

7 <title>Update</title>

8 <link rel="stylesheet" type="text/css" href="style.css"/>

9 </head>

10 <body>

11 <wicket:extend>

12 <fieldset>

13 <legend>Vis/ret</legend>

14 <form action="#" wicket:id="form">

15 <table align="center" border="0">

16 <tbody>

17 <tr>

18 <td style="width:120px">Overskrift:</td>

19 <td>

20 <input wicket:id="title" type="text"

21 name="" value="" size="50" />

22 </td>

23 </tr>

24 <tr>

25 <td style="width:120px">Produkt:</td>

26 <td>

27 <input wicket:id="product" type="text"

28 disabled="diabled" name="" value="" size="50" />

29 </td>

30 </tr>

31 <tr>

32 <td>Type:</td>

33 <td><select wicket:id="type" name="">

34 <option>Nyt ønske</option>

35 <option>Fejl</option>

36 </select></td>

37 </tr>

38 <tr>

39 <td>Oprettet af:</td>

40 <td><input wicket:id="user" type="text" name="" value="" size="50"

41 disabled="true"/></td>

42 </tr>

43 <tr>

44 <td>Oprettet den:</td>

45 <td><input wicket:id="created" type="text" name="" value=""

46 size="16" disabled="true" /></td>

47 </tr>

48 <tr>

49 <td>Afsluttet den:</td>

50 <td><input wicket:id="finished" type="text" name="" value=""

51 size="16" disabled="true"/></td>

52 </tr>

53 <tr>

54 <td>Sag status:</td>

55 <td><select wicket:id="caseStatus" name="">

56 <option>Ny</option>

57 <option>Under behandling</option>

58 <option>Godkendt</option>

59 <option>Afvist</option>

60 </select></td>

61 </tr>

62 <tr>

63 <td>Udviklings status:</td>

64 <td><select wicket:id="devStatus" name="">

65 <option>Ikke startet</option>

66 <option>Igang</option>

67 <option>Klar til test</option>

68 <option>I produktion</option>

69 </select></td>

70 </tr>

71 <tr>

72 <td>Beskrivelse:</td>

73 <td>&nbsp;</td>

74 </tr>

75 <tr>

76 <td colspan="2"><textarea wicket:id="description" name=""

77 rows="4" cols="80">

78 </textarea></td>

79 </tr>

80 <tr>

81 <td>Konklusion:</td>

82 <td>&nbsp;</td>

83 </tr>

84 <tr>

85 <td colspan="2"><textarea wicket:id="conclusion" name=""

86 rows="4" cols="80">

87 </textarea></td>

88 </tr>

89 <tr>

90 <td colspan="2">

91 <a href="#" wicket:id="showDiscussion">Gå til diskussion</a>

92 <div style="float:right">

93 <input wicket:id="cancel" type="button" value="Fortryd" />

94 <input wicket:id="delete" type="button" value="Slet" />

95 <input wicket:id="save" type="submit" value="Gem" />

96 </div>

97 </td>

98 </tr>

99 </tbody>

100 </table>

101 </form>

102 </fieldset>

103 </wicket:extend>

104 </body>

105 </html>

106

107

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\Update.java |

1 package dk.jsh.itdiplom.dbsw.bari.wicket;

2

3 import dk.jsh.itdiplom.dbsw.bari.business.BariCaseBusiness;

4 import dk.jsh.itdiplom.dbsw.bari.domain.BariCase;

5 import dk.jsh.itdiplom.dbsw.bari.domain.BariUser;

6 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.CaseStatus;

7 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.DevStatus;

8 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.Type;

9 import dk.jsh.itdiplom.dbsw.bari.domain.Constants.UserRole;

10 import java.util.ArrayList;

11 import java.util.List;

12 import org.apache.wicket.AttributeModifier;

13 import org.apache.wicket.Page;

14 import org.apache.wicket.markup.html.form.Button;

15 import org.apache.wicket.markup.html.form.DropDownChoice;

16 import org.apache.wicket.markup.html.form.Form;

17 import org.apache.wicket.markup.html.form.TextArea;

18 import org.apache.wicket.markup.html.form.TextField;

19 import org.apache.wicket.markup.html.link.Link;

20 import org.apache.wicket.model.Model;

21 import org.hibernate.StaleObjectStateException;

22

23 /\*\*

24 \* BariCase update page.

25 \*

26 \* @author Jan S. Hansen

27 \*/

28 public final class Update extends BasePage {

29 private TextField<String> title;

30 private TextField<String> product;

31 private DropDownChoice<String> type;

32 private DropDownChoice<String> caseStatus;

33 private DropDownChoice<String> devStatus;

34 private TextArea<String> description;

35 private TextArea<String> conclusion;

36

37 /\*\*

38 \* Constructor.

39 \*

40 \* @param bariCase A BaRI case.

41 \*/

42 public Update(final BariCase bariCase) {

43

44 //Add a form.

45 Form form = new Form("form") {

46 //Handle required fields errors.

47 @Override

48 protected void onError() {

49 List<String> emptyFields = new ArrayList<String>();

50 if (!title.checkRequired()) {

51 emptyFields.add("Overskrift");

52 title.add(new AttributeModifier("style", true,

53 new Model("border-color:red;")));

54 }

55 else {

56 title.add(new AttributeModifier("style", true,

57 new Model("border-color:default;")));

58 }

59 if (!type.checkRequired()) {

60 emptyFields.add("Type");

61 type.add(new AttributeModifier("style", true,

62 new Model("border-color:red;")));

63 }

64 else {

65 type.add(new AttributeModifier("style", true,

66 new Model("border-color:default;")));

67 }

68 if (!description.checkRequired()) {

69 emptyFields.add("Beskrivelse");

70 description.add(new AttributeModifier("style", true,

71 new Model("border-color:red;")));

72 }

73 else {

74 description.add(new AttributeModifier("style", true,

75 new Model("border-color:defalut;")));

76 }

77 StringBuilder errorMessage = new StringBuilder();

78 if (emptyFields.size() == 1) {

79 errorMessage.append("Feltet ");

80 errorMessage.append("'").append(emptyFields.get(0))

81 .append("'");

82 }

83 else {

84 errorMessage.append("Felterne ");

85 int fieldCounter = 1;

86 for (String field : emptyFields) {

87 errorMessage.append("'").append(field).append("'");

88 if (fieldCounter < emptyFields.size() -1) {

89 errorMessage.append(", ");

90 }

91 if (fieldCounter == emptyFields.size() -1) {

92 errorMessage.append(" og ");

93 }

94 fieldCounter++;

95 }

96 }

97 errorMessage.append(" skal udfyldes.");

98 setErrorMessage(errorMessage.toString());

99 }

100 };

101 add(form);

102

103 //Add form fields.

104 title = new TextField("title", new Model(bariCase.getTitle()));

105 title.setRequired(true);

106 form.add(title);

107 product = new TextField("product",

108 new Model(bariCase.getProduct().getName()));

109 form.add(product);

110 type = new DropDownChoice("type",

111 new Model(bariCase.getType().getDescription()),

112 Type.getDescriptions());

113 type.setRequired(true);

114 form.add(type);

115 form.add(new TextField("user",

116 new Model(bariCase.getBariUser().getFullname())));

117 form.add(new TextField("created",

118 new Model(standardDateTimeFormat.format(bariCase.getCreated()))));

119 form.add(new TextField("finished",

120 new Model(bariCase.getFinished() == null ? ""

121 :standardDateTimeFormat.format(bariCase.getFinished()))));

122 caseStatus = new DropDownChoice("caseStatus",

123 new Model(bariCase.getCaseStatus().getDescription()),

124 CaseStatus.getDescriptions());

125 caseStatus.setRequired(true);

126 form.add(caseStatus);

127 devStatus = new DropDownChoice("devStatus",

128 new Model(bariCase.getDevStatus().getDescription()),

129 DevStatus.getDescriptions());

130 devStatus.setRequired(true);

131

132 //Only ADMIN and DEVELOPER can change dev. status.

133 BariSession bariSession = BariSession.get();

134 BariUser currentBariUser = bariSession.getBariUser();

135 if (currentBariUser.getUserRole().equals(UserRole.NORMAL)) {

136 devStatus.setEnabled(false);

137 }

138

139 form.add(devStatus);

140 description = new TextArea("description",

141 new Model(bariCase.getDescription())) ;

142 description.setRequired(true);

143 form.add(description);

144 conclusion = new TextArea("conclusion",

145 new Model(bariCase.getConclusion()));

146 form.add(conclusion);

147

148 //Add form links and buttons

149 form.add(new Link("showDiscussion") {

150 @Override

151 public void onClick() {

152 Page page = new Discussion(bariCase);

153 setResponsePage(page);

154 }

155 });

156

157 form.add(new Link("cancel") {

158 @Override

159 public void onClick() {

160 setResponsePage(Overview.class);

161 }

162 });

163

164 Button saveButton = new Button("save") {

165 @Override

166 public void onSubmit() {

167 bariCase.setTitle(title.getModelObject());

168 bariCase.setType(Type.getType(type.getModelObject()));

169 bariCase.setCaseStatus(

170 CaseStatus.getCaseStatus(caseStatus.getModelObject()));

171 bariCase.setDevStatus(

172 DevStatus.getDevStatus(devStatus.getModelObject()));

173 bariCase.setDescription(description.getModelObject());

174 bariCase.setConclusion(conclusion.getModelObject());

175 try {

176 BariCaseBusiness.update(bariCase);

177 setResponsePage(Overview.class);

178 }

179 catch (StaleObjectStateException sose) {

180 setErrorMessage("Sagen kan ikke gemmes, " +

181 "da den er rettet af en anden!");

182 }

183 }

184 };

185 form.add(saveButton);

186 Link deleteLink = new Link("delete") {

187 @Override

188 public void onClick() {

189 BariCaseBusiness.delete(bariCase);

190 setResponsePage(Overview.class);

191 }

192 };

193 deleteLink.add(new JS\_EventConfirmation("onclick", "Er du sikker på" +

194 " at du vil slette?"));

195 form.add(deleteLink);

196

197 //Disable fields and save button, if case is finished.

198 if (bariCase.getFinished() != null) {

199 title.setEnabled(false);

200 type.setEnabled(false);

201 caseStatus.setEnabled(false);

202 devStatus.setEnabled(false);

203 description.setEnabled(false);

204 conclusion.setEnabled(false);

205 saveButton.setEnabled(false);

206 }

207 }

208

209 /\*\*

210 \* Inner class that adds a javascript confirm dialog to a attribute.

211 \*/

212 private class JS\_EventConfirmation extends AttributeModifier {

213

214 public JS\_EventConfirmation(String event, String msg) {

215 super(event, true, new Model(msg));

216 }

217

218 @Override

219 protected String newValue(final String currentValue,

220 final String replacementValue) {

221 String result = "if (confirm('" + replacementValue + "')) ";

222 if (currentValue != null) {

223 result += currentValue + ";";

224 }

225 return result;

226 }

227 }

228 }

229

|  |
| --- |
| \dk\jsh\itdiplom\dbsw\bari\wicket\style.css |

1 body {

2 margin:0; padding:0;

3 font-family:times;

4 }

5

6 div {

7 margin:0; padding:0; /\* Remove space between elements \*/

8 }

9

10 #container {

11 margin-left:auto; margin-right:auto; /\* Center block level content \*/

12 width:800px;

13 }

14

15 #header {

16 text-align: center; /\* Center inline Content - Don't work in IE'\*/

17 width:100%;

18 }

19

20 #content {

21 width:100%;

22 padding-left:10px; padding-right:10px;

23 }

24

25 #error {

26 text-align: center;

27 color: red;

28 }

29

30 #main {

31 width:100%;

32 }

33

34 /\* Table layout \*/

35 table.pageablelistview {

36 margin-bottom: 10px;

37 border-bottom: 1px solid;

38 }

39 table.pageablelistview caption {

40 text-align: left;

41 }

42 table.pageablelistview tr {

43 padding-top: 2px;

44 padding-bottom: 2px;

45 }

46 table.pageablelistview trtable.pageablelistview caption {

47 text-align: left;

48 }.even {

49 background-color: #ececec;

50 }

51 table.pageablelistview tr.odd {

52 background-color: #fff;

53 }

54 table.pageablelistview tr td {

55 padding-left: 8px; padding-right: 30px;

56 }

57 table.pageablelistview tr th {

58 color: black;

59 padding-top: 3px;

60 padding-bottom: 3px;

61 padding-left: 8px;

62 padding-right: 30px;

63 background-color: #ececec;

64 border-bottom: 1px solid;

65 border-top: 1px solid;

66 text-align: left;

67 white-space: nowrap;

68 vertical-align: middle;

69 }

70 table.pageablelistview tr th {

71 background-position: right;

72 background-repeat:no-repeat;

73 }

74 table.pageablelistview tr th a {

75 font-weight: normal;

76 }

77

### Hibernate configuration

|  |
| --- |
| hibernate.cfg.xml |

1 <?xml version="1.0" encoding="UTF-8"?>

2 <!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

3 <hibernate-configuration>

4 <session-factory>

5 <property name="hibernate.dialect">org.hibernate.dialect.DerbyDialect</property>

6 <property name="hibernate.connection.driver\_class">org.apache.derby.jdbc.ClientDriver</property>

7 <property name="hibernate.connection.url">jdbc:derby://localhost:1527/BARI</property>

8 <property name="hibernate.connection.username">bari</property>

9 <property name="hibernate.connection.password">bari</property>

10

11 <!-- Show and print nice SQL on stdout -->

12 <property name="hibernate.show\_sql">true</property>

13 <property name="hibernate.format\_sql">true</property>

14

15 <!-- List of annotated classes -->

16 <mapping class="dk.jsh.itdiplom.dbsw.bari.domain.Product" />

17 <mapping class="dk.jsh.itdiplom.dbsw.bari.domain.BariUser" />

18 <mapping class="dk.jsh.itdiplom.dbsw.bari.domain.UserGroup" />

19 <mapping class="dk.jsh.itdiplom.dbsw.bari.domain.BariCase" />

20 <mapping class="dk.jsh.itdiplom.dbsw.bari.domain.DiscussionMessage" />

21

22

23 </session-factory>

24 </hibernate-configuration>

25

26

## SQL til oprettelse og initering af databasen

Følgende fil er dannet af Hibernate til oprettelse af databasen.

|  |
| --- |
| bari\_ddl.sql |

1

2 alter table BariCase

3 drop constraint fk\_from\_baricase\_to\_product;

4

5 alter table BariCase

6 drop constraint fk\_from\_baricase\_to\_bariuser;

7

8 alter table DiscussionMessage

9 drop constraint fk\_from\_discussionmessage\_to\_baricase;

10

11 alter table DiscussionMessage

12 drop constraint fk\_from\_discussiommessage\_to\_bariuser;

13

14 alter table UserGroup

15 drop constraint fk\_from\_usergroup\_to\_product;

16

17 alter table UserGroup

18 drop constraint fk\_from\_usergroup\_to\_bariuser;

19

20 drop table BariCase;

21

22 drop table BariUser;

23

24 drop table DiscussionMessage;

25

26 drop table Product;

27

28 drop table UserGroup;

29

30 create table BariCase (

31 id bigint not null generated always as identity,

32 caseStatus varchar(15) not null,

33 conclusion varchar(400),

34 created timestamp not null,

35 description varchar(400) not null,

36 devStatus varchar(15) not null,

37 finished timestamp,

38 title varchar(50) not null,

39 type varchar(10) not null,

40 version integer not null,

41 bariUser\_id bigint not null,

42 product\_id bigint not null,

43 primary key (id)

44 );

45

46 create table BariUser (

47 id bigint not null generated always as identity,

48 fullname varchar(50) not null,

49 login varchar(20) not null unique,

50 password varchar(20) not null,

51 userRole varchar(10) not null,

52 version integer not null,

53 primary key (id)

54 );

55

56 create table DiscussionMessage (

57 id bigint not null generated always as identity,

58 created timestamp not null,

59 message varchar(400) not null,

60 version integer not null,

61 bariCase\_id bigint not null,

62 bariUser\_id bigint not null,

63 primary key (id)

64 );

65

66 create table Product (

67 id bigint not null generated always as identity,

68 name varchar(50) not null unique,

69 version integer not null,

70 primary key (id)

71 );

72

73 create table UserGroup (

74 id bigint not null generated always as identity,

75 version integer not null,

76 bariUser\_id bigint not null,

77 product\_id bigint not null,

78 primary key (id),

79 unique (bariUser\_id, product\_id)

80 );

81

82 alter table BariCase

83 add constraint fk\_from\_baricase\_to\_product

84 foreign key (product\_id)

85 references Product;

86

87 alter table BariCase

88 add constraint fk\_from\_baricase\_to\_bariuser

89 foreign key (bariUser\_id)

90 references BariUser;

91

92 alter table DiscussionMessage

93 add constraint fk\_from\_discussionmessage\_to\_baricase

94 foreign key (bariCase\_id)

95 references BariCase;

96

97 alter table DiscussionMessage

98 add constraint fk\_from\_discussiommessage\_to\_bariuser

99 foreign key (bariUser\_id)

100 references BariUser;

101

102 alter table UserGroup

103 add constraint fk\_from\_usergroup\_to\_product

104 foreign key (product\_id)

105 references Product;

106

107 alter table UserGroup

108 add constraint fk\_from\_usergroup\_to\_bariuser

109 foreign key (bariUser\_id)

110 references BariUser;

111

112

Følgende fil er oprettet for at tilføje ekstra constraints til database, samt oprette 3 produkter og 3 brugere.

|  |
| --- |
| bari\_init.sql |

1 -- Extra constrains

2 alter table bariuser

3 add constraint valid\_user\_roles

4 check (userrole in ('ADMIN', 'DEVELOPER', 'NORMAL'));

5

6 alter table bariuser

7 add constraint password\_length\_ge\_3

8 check (length(password) >= 3);

9

10 alter table baricase

11 add constraint valid\_type

12 check (type in ('ERROR', 'REQUEST'));

13

14 alter table baricase

15 add constraint valid\_dev\_stauts

16 check (devstatus in ('NOTSTARTED', 'STARTED', 'READYTOTEST', 'TESTED',

17 'INPRODUCTION'));

18

19 alter table baricase

20 add constraint valid\_case\_stauts

21 check (casestatus in ('NEW', 'CONSIDERING', 'APPROVED', 'REJECTED',

22 'DONE'));

23

24 -- Initialize database with some bari users.

25 insert into bariuser

26 (fullname, login, password, userrole, version)

27 values ('Jan Schrøder Hansen', 'jsh', 'jsh', 'ADMIN', 1);

28

29 insert into bariuser

30 (fullname, login, password, userrole, version)

31 values ('Kaj Kode Nørd', 'kkn', 'kkn', 'DEVELOPER', 1);

32

33 insert into bariuser

34 (fullname, login, password, userrole, version)

35 values ('Tanja Kikkenborg', 'tki', 'tki', 'NORMAL', 1);

36

37 -- Add some products and user groups

38 insert into product

39 (name, version)

40 values ('BaRI', 1);

41

42 insert into product

43 (name, version)

44 values ('Hibernate', 1);

45

46 insert into product

47 (name, version)

48 values ('Wicket', 1);

49

50 insert into usergroup

51 (bariuser\_id, product\_id, version)

52 values (1, 1, 1);

53

54 insert into usergroup

55 (bariuser\_id, product\_id, version)

56 values (1, 2, 1);

57

58 insert into usergroup

59 (bariuser\_id, product\_id, version)

60 values (1, 3, 1);

61

62 insert into usergroup

63 (bariuser\_id, product\_id, version)

64 values (2, 1, 1);

65

66 insert into usergroup

67 (bariuser\_id, product\_id, version)

68 values (2, 2, 1);

69

70 insert into usergroup

71 (bariuser\_id, product\_id, version)

72 values (2, 3, 1);

73

74 insert into usergroup

75 (bariuser\_id, product\_id, version)

76 values (3, 1, 1);

77

78 insert into usergroup

79 (bariuser\_id, product\_id, version)

80 values (3, 2, 1);

81

82 insert into usergroup

83 (bariuser\_id, product\_id, version)

84 values (3, 3, 1);

85

86

## Indhold på vedlagte CD

Indholdet på den vedlagte CD er inddelt i følgende 3 kataloger:

* Løsning – Indeholder java kode, html filer m.m samt NetBeans projektfil.
* Program – Indeholder en bari.war samt JavaDB skemafiler til oprettelse af databasen.
* Rapport – Indeholder denne rapport i Word 2007 og PDF format og diagrammer i Dia-format. Samt rapporten til faget Web og serverprogrammering, som denne opgave bygger videre på i PDF format.

1. Wicket – se <http://wicket.apache.org/> [↑](#footnote-ref-2)
2. Java – se <http://www.java.com/en/> [↑](#footnote-ref-3)
3. JavaDB – se <http://www.oracle.com/technetwork/java/javadb/overview/index.html> [↑](#footnote-ref-4)
4. Hibernate – se <http://www.hibernate.org/> [↑](#footnote-ref-5)
5. ER diagram – se <http://en.wikipedia.org/wiki/Er_diagram>, samt kap. 2 i Database Management Systems. [↑](#footnote-ref-6)
6. Referential integrity – se <http://en.wikipedia.org/wiki/Referential_integrity> [↑](#footnote-ref-7)
7. Optimistisk låsning – se <http://en.wikipedia.org/wiki/Optimistic_concurrency_control> [↑](#footnote-ref-8)
8. JDCB – se <http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136101.html> [↑](#footnote-ref-9)
9. query execution plan – se <http://en.wikipedia.org/wiki/Query_plan> [↑](#footnote-ref-10)
10. Enkelt undtagelse: En alm. bruger kan ikke ændre ”Udvikler status” feltet (devStatus) på BariCase tabellen. [↑](#footnote-ref-11)
11. Use case - <http://da.wikipedia.org/wiki/Use_case> [↑](#footnote-ref-12)
12. ULM – se <http://en.wikipedia.org/wiki/Unified_Modeling_Language> [↑](#footnote-ref-13)
13. Netbeans – se <http://netbeans.org/> [↑](#footnote-ref-14)
14. Apache Tomcat – se <http://tomcat.apache.org/> [↑](#footnote-ref-15)
15. SubVersion – se <http://subversion.tigris.org/> [↑](#footnote-ref-16)
16. GoogleCode – se <http://code.google.com/> [↑](#footnote-ref-17)
17. Model-View-Controller se - [http://en.wikipedia.org/wiki/Model-View-Controller](http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller) [↑](#footnote-ref-18)