IT-DIPLOMUDDANNELSEN

WEB OG SERVERPROGRAMMERING

Eksamensprojekt

af

Jan Schrøder Hansen

eFTERÅR 2009

**INDHOLD**

[1. Indledning 3](#_Toc294348289)

[2. Opgavebeskrivelse 3](#_Toc294348290)

[2.1. Afgrænsing 3](#_Toc294348291)

[3. Lidt om WICKET 4](#_Toc294348292)

[4. Design 5](#_Toc294348293)

[4.1. Designklassediagram 5](#_Toc294348294)

[4.2. Indeling af kode i lag 6](#_Toc294348295)

[5. Programmering 7](#_Toc294348296)

[6. Sikkerhed 8](#_Toc294348297)

[7. Brugervejledning 9](#_Toc294348298)

[8. Konklusion 15](#_Toc294348299)

[9. Bilag 16](#_Toc294348300)

[9.1. Fejlsituationer 16](#_Toc294348301)

[9.2. Kode 18](#_Toc294348302)

[9.2.1. Pakke dk.jsh.itdiplom.wsp.bari.wicket 18](#_Toc294348303)

[9.2.2. Pakke dk.jsh.itdiplom.wsp.bari.bussiness 34](#_Toc294348304)

[9.2.3. Pakke dk.jsh.itdiplom.wsp.bari.domain 36](#_Toc294348305)

[9.2.4. Pakke dk.jsh.itdiplom.wsp.bari.util 41](#_Toc294348306)

[9.2.5. Pakke dk.jsh.itdiplom.wsp.bari.wicket.test 42](#_Toc294348307)

[9.3. SQL til dannelse af databasen 43](#_Toc294348308)

[9.4. Konfigutationsfiler 43](#_Toc294348309)

[9.5. Ant target til generering af databaseskema 44](#_Toc294348310)

[9.6. Eksempler på unit test 45](#_Toc294348311)

[9.7. Indhold på vedlagte CD 45](#_Toc294348312)

# Indledning

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

Faget har taget udgangspunkt i bogen Webprogrammering med JSP af Jacob Nordfalk, som også er underviser i faget.

# 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.

Et andet mål med opgaven er at afprøve web-frameworket Wicket[[1]](#footnote-2). Jeg har taget udgangspunkt i bogen Wicket in Action[[2]](#footnote-3) af Martijn Dashorst og Eelco Hillenius.

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

## Afgrænsing

For at afgrænse opgaven har jeg valgt, at se bort fra login, brugerroller og fil uploads. Det fremgår også af mit designklassediagram, se [Designklassediagram](#_Designklasse_diagram), hvilke klasser som er implementeret.

Strenge og dato/tid formater er skrevet direkte i koden, så hvis programmet skal benyttes på andre sprog end dansk, skal alle strenge og dato/tid formater m.m., flyttes ud i nogle ressource filer.

Løsningen er heller ikke dækket ind med hensyn til unit tests. Der er dog lavet en enkelt unit test, for at demonstrere at Wicket er forberedt til unit tests.

# Lidt om WICKET

På Wickets hjemmeside kan man læse om de mål, udviklerne har haft med Wicket[[3]](#footnote-4). Jeg vil her komme ind på de punkter, jeg selv som udvikler lægger vægt på.

Som udvikler har jeg arbejdet med Struts 1.3, samt med ASP.NET, og en af de problemer som er i disse frameworks, er bl.a. at man blander kode med html.

I Struts arbejdes der med java, jsp tags, html tags, javascript i en og samme fil. Det samme gør sig også gældende i ASP.NET, hvor det så bare er C# kode, ASP tags, html tags. Og det giver nogle filer som er svære at overskue, svære at genbruge og vedligeholde.

Helt grundlæggende prøver Wicket, at adskille html og kode. Det giver mulighed for, at det faktisk er muligt at få en html/css specialist, til at lave selve html/css koden. Som så kan overtages af en programmør. Det er ikke rigtigt muligt med f.eks struts og ASP.NET, da der er så mange specielle tags, som en html specialist ikke kender til, og som ikke kan håndteres af webdesigners værktøjer.

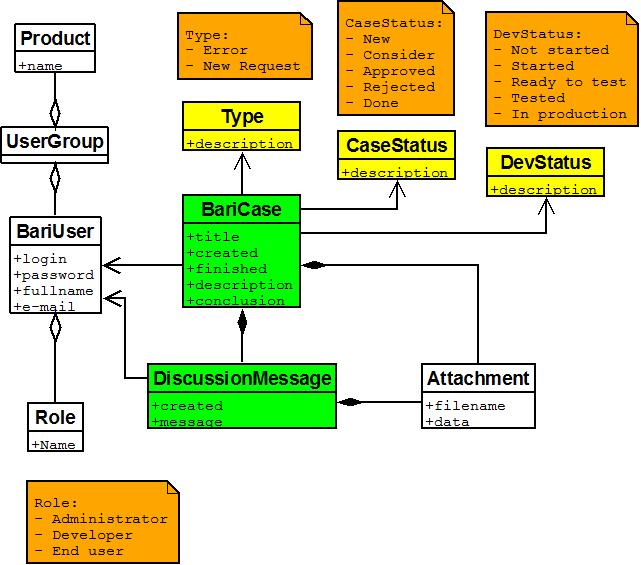
Wicket benytter også nogle gamle java dyder som, en klasse, en java fil. I Wicket kan en Wicket webside, beskrives vha. en html fil og en java fil. Begge ligger i samme katalog og hedder det samme, på nær fil endelsen. Det eneste krav Wicket stiller til html filerne er, at de html elementer der skal være dynamiske, skal have en entydigt Wicket identifikation.

En anden ting med Wicket er, at Unit test er tænkt ind i systemet fra starten af. Det er nemt at tage en Wicket side, og teste den vha. JUnit. Se Bilag [Eksempler på unit test](#_Eksempler_på_unit).

# Design

Følgende er en kort beskrivelse af designet af BaRI webapplikationen. Startende med et designklassediagram.

## Designklassediagram



Figur 1 - Designklassediagram

De hvide klasser er ikke implementeret endnu. BariUser er delvist implementeret, den persisteres dog ikke i databasen.

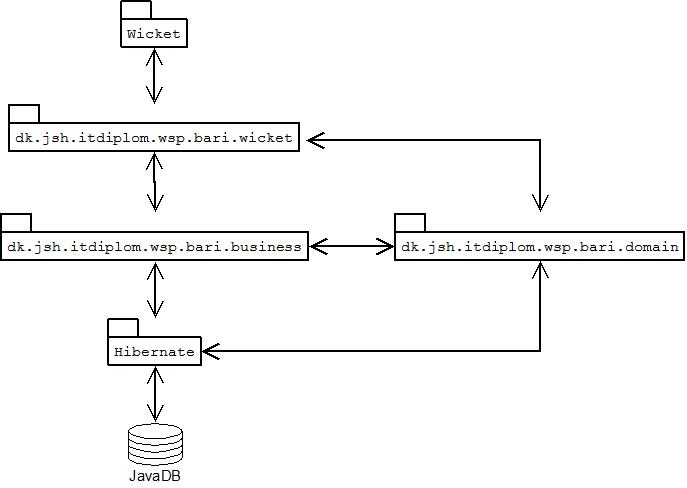
De grønne klasser er fuldt implementeret, dvs. at de også persisteres. De gule klasser er implementeret som java enums. Dvs. at deres værdi bliver persisteret sammen med BariCase.

Det er meningen at BaRI skal kunne håndtere forskellige software projekter/produkter, det er derfor, der er en Product klasse og en UserGroup klasse, så man kan knytte brugere til de enkelte produkter. De enkelte brugere skal også kunne have forskellige roller, som styres af Role klassen.

Klassen BariUser og Role kunne evt. håndteres af Webserveren, der som standard kan tilbyde disse funktioner, enten vha. Webserveres konfigurationsfiler. Webserveren kan også konfigureres så den henter brugere og roller fra en database.

## Indeling af kode i lag

Jeg har inddelt min kode i 3 overordnet lag, vha. java pakker, se følgende diagram.



Figur 2 – Pakkediagram

Pakkerne som starter med ”dk.jsh.itdiplom.wsp.bari”, er udviklet i forbindelse med denne opgave. Wicket og Hibernate[[4]](#footnote-5) er de to frameworks som benyttes til henholdsvis Web delen og til database delen. JavaDB er den database som benyttes.

De 3 pakker som starter med ”dk.jsh.itdiplom.wsp.bari” indeholder følgende:

|  |  |
| --- | --- |
| **Pakke** | **Beskrivelse** |
| dk.jsh.itdiplom.bari.wicket | Kode til generering af de enkelte web-sider. |
| dk.jsh.itdiplom.wsp.bari.busines | Forretningslogik, i forbindelse med Hibernate |
| dk.jsh.idtiplom.wsp.bari.domain | Data elementer, som dels persisteres, og som benyttes som value objects, mellem de andre lag. De samme klasser som fremgår af Designklassediagrammet. |

Hibernate benyttes sammen med domain klasserne, som har annotations på de attributter som skal persisteres. Ud fra disse kan Hibernate vha. et ant target lave en database skemafil og håndtere adgang til databasen.

Derudover er der en enkel pakke mere som benyttes til Unit test[[5]](#footnote-6), pakken hedder dk.jsh.itdiplom.wsp.bari.wicket.test.

# Programmering

Under programmeringen har jeg gjort brug af følgende værktøjer og teknikker:

* NetBeans[[6]](#footnote-7) – Java IDE, Java udviklingsmiljø.
* JavaDB[[7]](#footnote-8) – tidligere Derby, er nu en del af standard java.
* Apache Tomcat[[8]](#footnote-9) – Webserver.
* SubVersion[[9]](#footnote-10) - Repository til al kode. Bl.a for at have backup af koden på en anden maskine, og for at få historik på mine koderettelser.
* Frameworks: Hibernate og Wicket, beskrevet tidligere.

Alle diagrammer er lavet vha. Dia[[10]](#footnote-11), som har skabeloner til UML diagrammer. De enkelte diagrammer ligger også på den vedlagte CD. Se [bilag](#_Indhold_på_vedlagte) for indholdet på den vedlagte CD.

# Sikkerhed

Wicket er ifølge wickets hjemmeside[[11]](#footnote-12) ”secure by default”. Og jeg har heller ikke kunne fremprovokere diverse ”Injection flaws”[[12]](#footnote-13) angreb.

SQL injections forebygger jeg vha. Hibernate’s måde at lave prepared statement på, f.eks som i følgende kode:

String hql = "select discussionMessage from "

+ "dk.jsh. . .wsp.bari.domain.DiscussionMessage discussionMessage "

+ "where bariCase.id = :id "

+ "order by discussionMessage.created";

Query query = session.createQuery(hql);

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

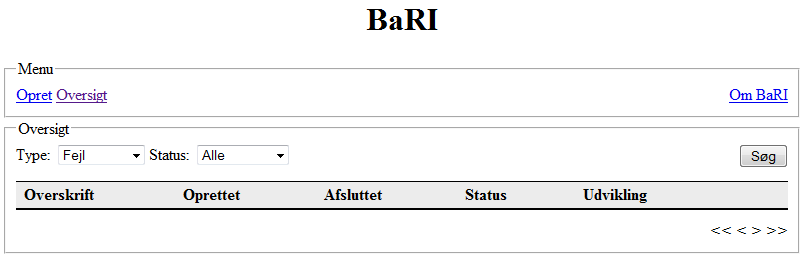
Som det fremgår af koden, bliver ingen variabler direkte indsat i SQL strengen, men via metoden setString().

Med hensyn til adgangskontrol så har data modellen, se [Designklassediagram](#_Designklasse_diagram), en bruger (BariUser), med nogle roller (Role). Dette ligger tæt op af, hvad en Webserver tilbyder. Så jeg vil anbefale at man bruger det indbyggede brugeradgangskontrol, som findes i webserveren, også kaldet containerstyret adgangskontrol, sammen med HTTPS/SSL[[13]](#footnote-14), hvis man vil være helt sikker på, at et evt. password ikke bliver opsnappet. Men derudover mener jeg ikke, at der er noget i denne Webapplikation, som er så følsomt at det kræver HTTPS.

Hvis man ikke vil betale for et HTTPS/SSL certifikat, kan man nøjes med at benytte Auth-method: Digest, som ikke overføre password i klartekst, men som en hashværdi. Det beskytter dog ikke mod man-in-the-middle[[14]](#footnote-15) angreb.

# Brugervejledning

Første gang BaRI startes vil følgende tomme oversigts side vises.



Figur 3 – Oversigtsside

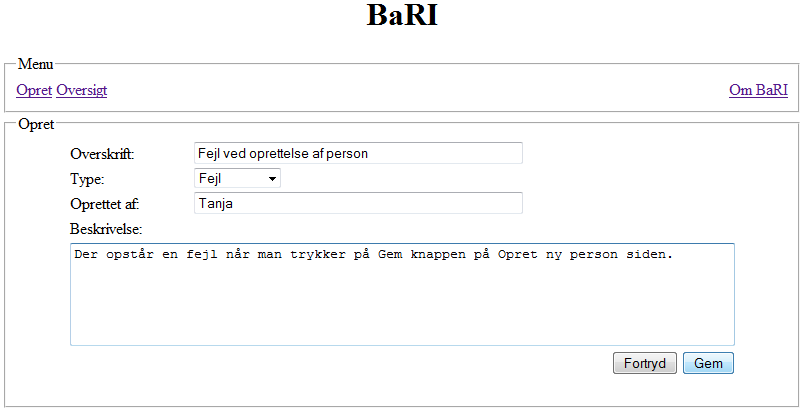
BaRI er opdelt i to områder. Et Menu område som benyttes til at navigere rundt i BaRI med. Denne er ens for alle BaRI sider. Derunder er der et arbejdsområde, som skiftes ud efter de valg der fortages i Menu området.

For at oprette en ny sag, trykkes der på ”Opret” linket i menu boksen og følgende skærmbillede fremkommer.



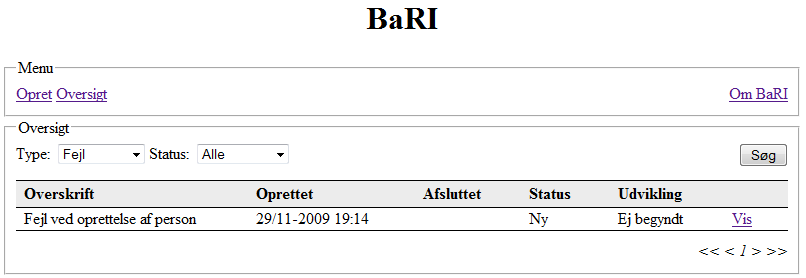
Figur 4 - Opret ny sag side

På denne side kan man oprette enten en ny fejl eller et nyt ønske. Det bestemmes af ”Type” feltet. På det næste billede er man klar til at gemme en ny fejl.



Figur 5 - Oprettelse af en fejl

Når der trykkes på ”Gem” knappen, så gemmes sagen og man kommer tilbage til oversigtssiden. Fortryd returnerer også til oversigtssiden, dog uden at gemme.



Figur 6 - Retur til oversigtssiden

Øverst over listen kan man ændre søgekriterierne for listen. Man kan vælge at se enten alle fejl eller alle ønsker. Man kan også søge efter hvilken status, de enkelte sager har vha. ”Status” feltet. Hvis status ”Alle” vælges, så vises alle sager af den valgte type. Listen opdateres ved at trykke på ”Søg” knappen.

Til højre under listen ses følgende navigationslinks:



|  |  |
| --- | --- |
| **Link** | **Beskrivelse** |
| << | Viser første 10 sager i listen. |
| < | Viser forrige 10 sager. |
| 1 | Går direkte til side, her 1. |
| > | Viser de næste 10 sager. |
| >> | Viser de sidste sager i listen. |

Nu kan den nye fejl vedligeholdes ved at trykke på ”Vis” linket i listen. Og følgende side vises.



Figur 7 - Side for vedligeholdelse af en sag

Felterne ”Oprettet af”, ”Oprettet den” og ”Afsluttet den” kan ikke rettes.

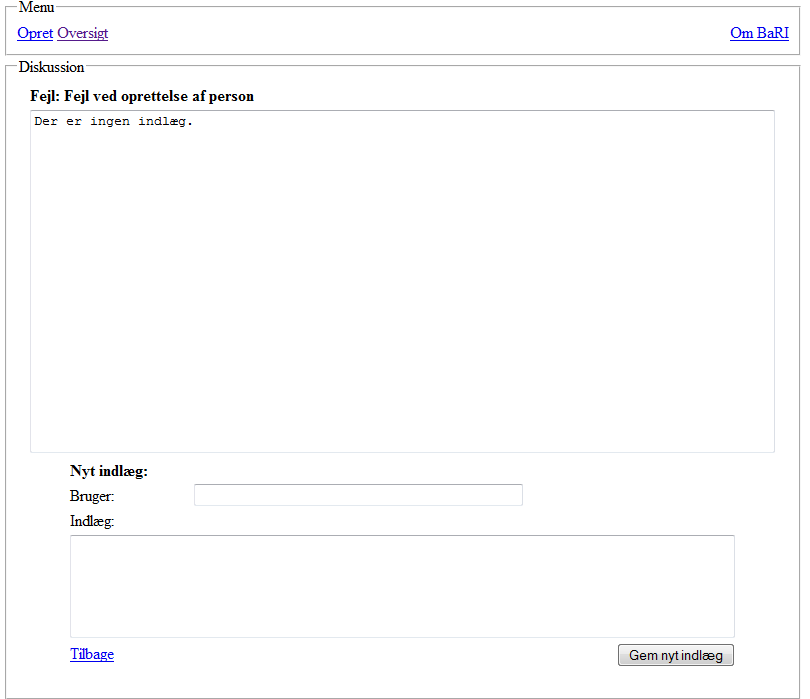
”Sags status” kan have følgende værdier:

|  |  |
| --- | --- |
| **Værdi** | **Beskrivelse** |
| Ny | Det er en ny sag. |
| Behandles | Sagen behandles, dvs. der bliver taget stilling til, om man skal gå videre med sagen. |
| Godkendt | Sagen er godkendt og dermed klar til en udvikler. |
| Afvist | Sagen er afvist. |
| Afsluttet | Sagen er færdigudviklet og afsluttet. Feltet ”Afsluttet den” får dagsdato. |

”Udviklings status” kan have følgende værdier:

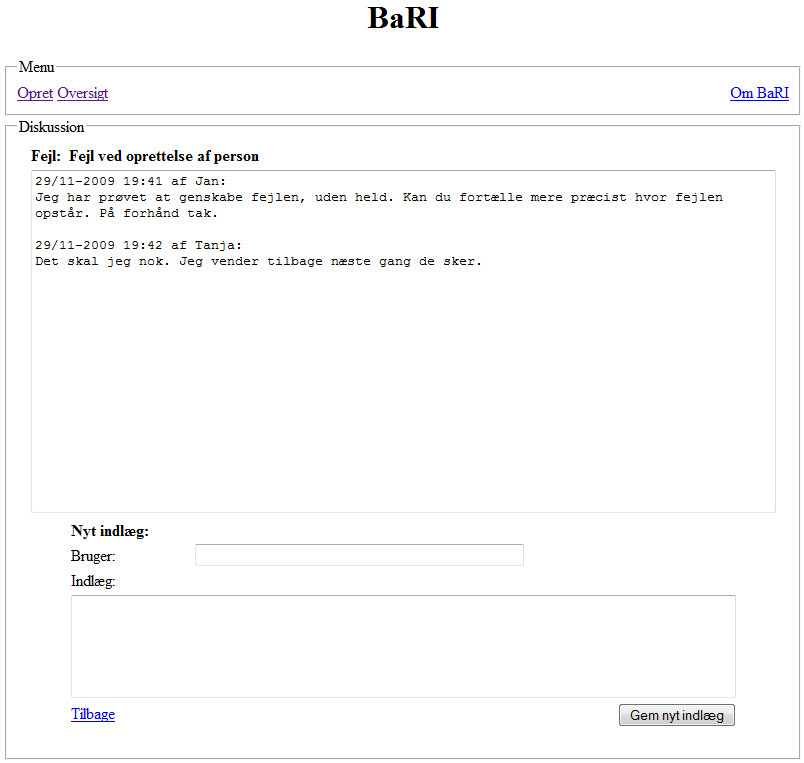
|  |  |
| --- | --- |
| **Værdi** | **Beskrivelse** |
| Ej begyndt | Venter på en udvikler. |
| I gang | Sagen er i gang. |
| Klar til test | Sagen venter på at blive testet. |
| Testet | Sagen er testet. |
| I prod. | Sagen er i produktion |

Der er mulighed for at rette oplysninger, slette hele sagen, vha. af ”Gem” og ”Slet” knapperne. Eller man kan gå til en diskussions side ved at trykke på ”Gå til diskussion” linket. Som vises på følgende side.



Figur 8 - Diskussionsside

Denne side giver mulighed for, at slutbrugere og udviklere kan diskutere forskellige ting. F.eks. løsningsforslag til et nyt ønske. Følgende side er et eksempel på en diskussion.



Figur 9 - Eksempel på en diskussion

Til slut - ”Om BaRI” linket giver følgende side:



Figur 10 - Om BaRI siden

Se også [Fejlsituationer](#_Fejl_sitationer), under bilag for eksempler på fejl situationer i BaRI.

# Konklusion

Det har været en god oplevelse at arbejde med Wicket, man kommer meget hurtigt i gang. Og jeg synes at Wicket lever meget fint op til de forventninger, jeg havde til frameworket. Der er en meget klar adskillelse mellem html og java. Det ses bl.a. meget tydeligt på de små og meget overskuelige html-sider i dette projekt. Wicket gør også meget for at gemme webapplikations-problematikkerne væk fra programmøren. Så denne kan koncentrere sig som selve forretningslogikken.

Og det var dejligt at sidde og kode i java igen. Html delen er hurtigt overstået med Wicket.

Wicket arbejder også godt sammen med andre frameworks, jeg har benyttet Hibernate til database persistense, uden problemer.

Hvis jeg skulle nævne en enkelt bekymring, så skulle det måske være at Wicket bruger en del hukommelse på server siden. Da Wicket bruger de enkelte webserverobjekter som session request meget. Så jeg er ikke sikker på at Wicket egner sig til systemer, som rammes af tusindvis af brugere dagligt.

Men ellers vil jeg varmt anbefale Wicket.

# Bilag

## Fejlsituationer

Følgende er eksempler på fejlsituationer i BaRI. Første eksempel er et forsøg på at gemme en sag, hvor feltet ”Beskrivelse” ikke er udfyldt.



Figur 11 - Obligatorisk felt mangler

Andet eksempel er et forsøg på at rette en sag, som er rettet af en anden person i mellemtiden. BaRI benytter optimistisk låsning vha. Hibernate.



Figur 12 - Optimistisk låsning

## Kode

Koden er delt op de enkelte javapakker.

### Pakke dk.jsh.itdiplom.wsp.bari.wicket

#### Application.java

**package** dk.jsh.itdiplom.wsp.bari.wicket;

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

/\*\*

\* Wicket application.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** Application **extends** WebApplication {

/\*\*

\* Constructor.

\*/

**public** Application() {

}

/\*\*

\* Returns home page for the application.

\*/

@Override

**public** Class getHomePage() {

**return** Overview.**class**;

}

}

#### BasePage.html

Denne htmlside er en fælles side for alle sider, en slags abstract html side, som de andre sider arver fra. Det er her menuen og fejltekstområdet defineres.

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

<html>

<head>

<title>BaRI</title>

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

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

</head>

<body>

<div id=*"container"*>

<div id=*"header"*>

<h1>BaRI</h1>

</div>

<div id=*"contents"*>

<fieldset>

<legend>Menu</legend>

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

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

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

Om BaRI</a>

</fieldset>

</div>

<div id=*"error"*>

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

</div>

<div id=*"main"*>

<wicket:child />

</div>

</div>

</body>

</html>

#### BasePage.java

Det samme som beskrevet for BasePage.html gør sig gældende for denne fil.

**package** dk.jsh.itdiplom.wsp.bari.wicket;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariUser;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.Type;

**import** java.text.SimpleDateFormat;

**import** org.apache.wicket.Page;

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

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

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

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

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

/\*\*

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

\* and menu links.

\*

\* **@author** Jan S. Hansen

\*/

**public** **abstract** **class** BasePage **extends** WebPage {

**final** **static** SimpleDateFormat standardDateTimeFormat =

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

**private** String errorMessage = "";

**private** BariUser bariUser;

/\*\*

\* Constructor

\*/

**public** BasePage() {

PropertyModel errorMessageModel =

**new** PropertyModel(**this**, "errorMessage");

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

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

add(**new** Link("createnew") {

@Override

**public** **void** onClick() {

Page page = **new** CreateNew(bariUser);

setResponsePage(page);

}

});

add(**new** Link("overview") {

@Override

**public** **void** onClick() {

Page page = **new** Overview(bariUser, Type.ERROR, "Alle");

setResponsePage(page);

}

});

add(**new** Link("about") {

@Override

**public** **void** onClick() {

Page page = **new** About();

setResponsePage(page);

}

});

}

/\*\*

\* Get BaRI user.

\*/

**public** BariUser getBariUser() {

**return** bariUser;

}

/\*\*

\* Set BaRI user.

\*/

**public** **void** setBariUser(BariUser bariUser) {

**this**.bariUser = bariUser;

}

/\*\*

\* Returns an error message

\*/

**public** String getErrorMessage() {

**return** errorMessage;

}

/\*\*

\* Set error message.

\*/

**public** **void** setErrorMessage(String errorMessage) {

**this**.errorMessage = errorMessage;

}

}

#### About.html

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

<html>

<head>

<title></title>

</head>

<body>

<wicket:extend>

<fieldset>

<legend>Om BaRI</legend>

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

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

<p>Programmet er udviklet i forbindelse med faget Web og Server

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

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

efter&aring;ret 2009.</p>

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

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

</fieldset>

<fieldset>

<legend>M&aring;ling af kodekvalitet -

What the fuck per minute</legend>

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

</fieldset>

</wicket:extend>

</body>

</html>

#### About.java

**package** dk.jsh.itdiplom.wsp.bari.wicket;

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

/\*\*

\* About page.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** About **extends** BasePage {

**public** About() {

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

}

}

#### CreateNew.html

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

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

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

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

<head>

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

<title></title>

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

</head>

<body>

<wicket:extend>

<fieldset>

<legend>Opret</legend>

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

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

<tbody>

<tr>

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

<td>

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

</td>

</tr>

<tr>

<td>Type:</td>

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

<option>Nyt ønske</option>

<option>Fejl</option>

</select></td>

</tr>

<tr>

<td>Oprettet af:</td>

<td>

<input wicket:id=*"user"* type=*"text"* name=*""* value=*""* size=*"50"* />

</td>

</tr>

<tr>

<td>Beskrivelse:</td>

<td>&nbsp;</td>

</tr>

<tr>

<td colspan=*"2"*>

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

</textarea>

</td>

</tr>

<tr>

<td colspan=*"2"*>

<div style=*"float:right"*>

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

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

</div>

</td>

</tr>

</tbody>

</table>

</form>

</fieldset>

</wicket:extend>

</body>

</html>

#### CreateNew.java

**package** dk.jsh.itdiplom.wsp.bari.wicket;

**import** dk.jsh.itdiplom.wsp.bari.bussiness.BariCaseBusiness;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariCase;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariUser;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.CaseStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.DevStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.Type;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.List;

**import** org.apache.wicket.AttributeModifier;

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

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

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

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

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

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

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

/\*\*

\* Create new BariCase page.

\*

\* **@author** Jan S. Hansen

\*/

**public** **final** **class** CreateNew **extends** BasePage {

**private** TextField<String> title;

**private** DropDownChoice<String> type;

**private** TextField<String> user;

**private** TextArea<String> description;

/\*\*

\* Constructor.

\*

\* **@param** bariUser A BaRI user.

\*/

**public** CreateNew(BariUser bariUser) {

//Add a form as an inner class.

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

//Handles required fields error.

@Override

**protected** **void** onError() {

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

**if** (!title.checkRequired()) {

emptyFields.add("Overskrift");

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

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

}

**else** {

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

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

}

**if** (!type.checkRequired()) {

emptyFields.add("Type");

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

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

}

**else** {

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

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

}

**if** (!user.checkRequired()) {

emptyFields.add("Oprettet af");

user.add(**new** AttributeModifier("style", **true**,

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

}

**else** {

user.add(**new** AttributeModifier("style", **true**,

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

}

**if** (!description.checkRequired()) {

emptyFields.add("Beskrivelse");

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

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

}

**else** {

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

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

}

StringBuilder errorMessage = **new** StringBuilder();

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

errorMessage.append("Feltet ");

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

.append("'");

}

**else** {

errorMessage.append("Felterne ");

**int** fieldCounter = 1;

**for** (String field : emptyFields) {

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

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

errorMessage.append(", ");

}

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

errorMessage.append(" og ");

}

fieldCounter++;

}

}

errorMessage.append(" skal udfyldes.");

setErrorMessage(errorMessage.toString());

}

};

add(form);

//Add fields to the form.

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

title.setRequired(**true**);

form.add(title);

type = **new** DropDownChoice("type",

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

type.setRequired(**true**);

form.add(type);

user = **new** TextField("user", **new** Model(""));

user.setRequired(**true**);

form.add(user);

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

description.setRequired(**true**);

form.add(description);

//Add buttons to the form.

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

@Override

**public** **void** onClick() {

setResponsePage(Overview.**class**);

}

});

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

@Override

**public** **void** onSubmit() {

BariCase bariCase = **new** BariCase(

title.getModelObject(),

Type.getType(type.getModelObject()),

user.getModelObject(),

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

DevStatus.NOTSTARTED,

description.getModelObject(),

**null**);

BariCaseBusiness.saveNew(bariCase);

setResponsePage(Overview.**class**);

}

});

}

}

#### Discussion.html

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

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

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

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

<head>

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

<title>Discussion</title>

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

</head>

<body>

<wicket:extend>

<fieldset>

<legend>Diskussion</legend>

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

<tbody>

<tr>

<td colspan=*"2"*>

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

<span wicket:id=*"title"*>Overskrift</span></b>

</td>

</tr>

<tr>

<td colspan=*"2"*>

<textarea wicket:id=*"log"* name=*""* rows=*"20"* cols=*"90"*

readonly=*"readonly"*>

</textarea>

</td>

</tr>

</tbody>

</table>

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

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

<tbody>

<tr>

<td colspan=*"2"*>

<b>Nyt indlæg:</b>

</td>

</tr>

<tr>

<td style=*"width:120px"*>Bruger:</td>

<td>

<input wicket:id=*"user"* type=*"text"* name=*""* value=*""*

size=*"50"* />

</td>

</tr>

<tr>

<td colspan=*"2"*>Indlæg:</td>

</tr>

<tr>

<td colspan=*"2"*>

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

rows=*"5"* cols=*"80"*>

</textarea>

</td>

</tr>

<tr>

<td colspan=*"2"* >

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

<div style=*"float:right"*>

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

value=*"Gem nyt indlæg"* />

</div>

</td>

</tr>

</tbody>

</table>

</form>

</fieldset>

</wicket:extend>

</body>

</html>

#### Discussion.java

**package** dk.jsh.itdiplom.wsp.bari.wicket;

**import** dk.jsh.itdiplom.wsp.bari.bussiness.DiscussionMessageBusiness;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariCase;

**import** dk.jsh.itdiplom.wsp.bari.domain.DiscussionMessage;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.List;

**import** org.apache.wicket.AttributeModifier;

**import** org.apache.wicket.Page;

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

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

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

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

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

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

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

/\*\*

\* Discussion page.

\*

\* **@author** Jan S. Hansen

\*/

**public** **final** **class** Discussion **extends** BasePage {

**private** TextField<String> user;

**private** TextArea<String> message;

/\*\*

\* Constructor.

\*

\* **@param** bariCase A BaRI user.

\*/

**public** Discussion(**final** BariCase bariCase) {

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

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

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

List<DiscussionMessage> discussionMessages =

DiscussionMessageBusiness.getAllDiscussionMessages(bariCase);

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

StringBuilder log = **new** StringBuilder();

**for** (DiscussionMessage discussionMessage : discussionMessages) {

log.append(standardDateTimeFormat.format(

discussionMessage.getCreated()));

log.append(" af ");

log.append(discussionMessage.getBariUser());

log.append(":\n");

log.append(discussionMessage.getMessage());

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

}

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

}

**else** {

add(**new** TextArea("log", **new** Model("Der er ingen indlÃ¦g.")));

}

//Create and add a form

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

//Handles required fields error.

@Override

**protected** **void** onError() {

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

**if** (!user.checkRequired()) {

emptyFields.add("Bruger");

user.add(**new** AttributeModifier("style", **true**,

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

}

**else** {

user.add(**new** AttributeModifier("style", **true**,

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

}

**if** (!message.checkRequired()) {

emptyFields.add("IndlÃ¦g");

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

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

}

**else** {

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

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

}

StringBuilder errorMessage = **new** StringBuilder();

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

errorMessage.append("Feltet ");

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

.append("'");

}

**else** {

errorMessage.append("Felterne ");

**int** fieldCounter = 1;

**for** (String field : emptyFields) {

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

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

errorMessage.append(", ");

}

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

errorMessage.append(" og ");

}

fieldCounter++;

}

}

errorMessage.append(" skal udfyldes.");

setErrorMessage(errorMessage.toString());

}

};

add(form);

//Add fields to the form.

user = **new** TextField("user", **new** Model(""));

user.setRequired(**true**);

form.add(user);

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

message.setRequired(**true**);

form.add(message);

//Add links and buttons to the form.

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

@Override

**public** **void** onClick() {

Page page = **new** Update(bariCase);

setResponsePage(page);

}

});

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

@Override

**public** **void** onSubmit() {

DiscussionMessage discussionMessage = **new** DiscussionMessage(

bariCase, **new** Date(), user.getModelObject(),

message.getModelObject());

DiscussionMessageBusiness.saveNew(discussionMessage);

Page page = **new** Discussion(bariCase);

setResponsePage(page);

}

});

}

}

#### Overview.html

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

<html>

<head>

<title></title>

</head>

<body>

<wicket:extend>

<fieldset>

<legend>Oversigt</legend>

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

Type:&nbsp;

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

<option>Nyt Ã¸nske</option>

<option>Fejl</option>

</select>

Status:&nbsp;

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

<option>Alle</option>

<option>Ny</option>

<option>Under behandling</option>

<option>Godkendt</option>

<option>Afvist</option>

</select>

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

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

</form>

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

<tr>

<th>Overskrift</th>

<th>Oprettet</th>

<th>Afsluttet</th>

<th>Status</th>

<th>Udvikling</th>

<th>&nbsp;</th>

</tr>

<tr wicket:id=*"pageable"*>

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

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

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

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

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

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

</tr>

</table>

<div style=*"float:right"*>

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

</div>

</fieldset>

</wicket:extend>

</body>

</html>

#### Overview.java

**package** dk.jsh.itdiplom.wsp.bari.wicket;

**import** dk.jsh.itdiplom.wsp.bari.bussiness.BariCaseBusiness;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariCase;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariUser;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.CaseStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.Type;

**import** java.util.LinkedList;

**import** java.util.List;

**import** org.apache.wicket.AttributeModifier;

**import** org.apache.wicket.Page;

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

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

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

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

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

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

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

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

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

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

/\*\*

\* Overview page for all BariCases.

\*

\* **@author** Jan S. Hansen

\*/

**public** **final** **class** Overview **extends** BasePage {

**private** DropDownChoice<String> dropDownChoiceType;

**private** DropDownChoice<String> statusDownChoiceType;

//**TODO**: Should be deleted after autentication is implemented.

**public** Overview() {

**this**(**new** BariUser("login", "password", "fullname", "email"),

Type.ERROR, "Alle");

}

/\*\*

\* Constructor.

\*

\* **@param** bariUser A BaRI user.

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

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

\*/

**public** Overview(**final** BariUser bariUser, Type type, String status) {

setBariUser(bariUser);

//Add search form.

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

add(form);

dropDownChoiceType = **new** DropDownChoice("type",

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

form.add(dropDownChoiceType);

LinkedList<String> statusList =

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

statusList.addFirst("Alle");

statusDownChoiceType =

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

form.add(statusDownChoiceType);

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

@Override

**public** **void** onSubmit() {

Page page = **new** Overview(bariUser,

Type.getType(dropDownChoiceType.getModelObject()),

statusDownChoiceType.getModelObject());

setResponsePage(page);

}

});

//Add table with search results.

CaseStatus cs = **null**;

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

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

}

List<BariCase> bariCases = BariCaseBusiness.getAllBariCases(type, cs);

PageableListView pageableListView =

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

@Override

**protected** **void** populateItem(**final** ListItem item) {

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

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

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

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

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

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

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

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

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

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

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

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

@Override

**public** **void** onClick() {

Page page = **new** Update(bariCase);

setResponsePage(page);

}

});

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

**true**, **new** AbstractReadOnlyModel<String>()

{

@Override

**public** String getObject()

{

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

}

}));

}

};

add(pageableListView);

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

}

}

#### Update.html

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

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

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

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

<head>

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

<title>Update</title>

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

</head>

<body>

<wicket:extend>

<fieldset>

<legend>Vis/opdater</legend>

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

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

<tbody>

<tr>

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

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

</tr>

<tr>

<td>Type:</td>

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

<option>Nyt ønske</option>

<option>Fejl</option>

</select></td>

</tr>

<tr>

<td>Oprettet af:</td>

<td><input wicket:id=*"user"* type=*"text"* name=*""* value=*""* size=*"50"* disabled=*"true"*/></td>

</tr>

<tr>

<td>Oprettet den:</td>

<td><input wicket:id=*"created"* type=*"text"* name=*""* value=*""* size=*"16"* disabled=*"true"* /></td>

</tr>

<tr>

<td>Afsluttet den:</td>

<td><input wicket:id=*"finished"* type=*"text"* name=*""* value=*""* size=*"16"* disabled=*"true"*/></td>

</tr>

<tr>

<td>Sag status:</td>

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

<option>Ny</option>

<option>Under behandling</option>

<option>Godkendt</option>

<option>Afvist</option>

</select></td>

</tr>

<tr>

<td>Udviklings status:</td>

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

<option>Ikke startet</option>

<option>Igang</option>

<option>Klar til test</option>

<option>I produktion</option>

</select></td>

</tr>

<tr>

<td>Beskrivelse:</td>

<td>&nbsp;</td>

</tr>

<tr>

<td colspan=*"2"*><textarea wicket:id=*"description"* name=*""* rows=*"5"* cols=*"80"*>

</textarea></td>

</tr>

<tr>

<td>Konklusion:</td>

<td>&nbsp;</td>

</tr>

<tr>

<td colspan=*"2"*><textarea wicket:id=*"conclusion"* name=*""* rows=*"5"* cols=*"80"*>

</textarea></td>

</tr>

<tr>

<td colspan=*"2"*>

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

<div style=*"float:right"*>

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

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

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

</div>

</td>

</tr>

</tbody>

</table>

</form>

</fieldset>

</wicket:extend>

</body>

</html>

#### Update.java

**package** dk.jsh.itdiplom.wsp.bari.wicket;

**import** dk.jsh.itdiplom.wsp.bari.bussiness.BariCaseBusiness;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariCase;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.CaseStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.DevStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.Type;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.apache.wicket.AttributeModifier;

**import** org.apache.wicket.Page;

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

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

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

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

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

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

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

**import** org.hibernate.StaleObjectStateException;

/\*\*

\* BariCase update page.

\*

\* **@author** Jan S. Hansen

\*/

**public** **final** **class** Update **extends** BasePage {

**private** TextField<String> title;

**private** DropDownChoice<String> type;

**private** DropDownChoice<String> caseStatus;

**private** DropDownChoice<String> devStatus;

**private** TextArea<String> description;

**private** TextArea<String> conclusion;

/\*\*

\* Constructor.

\*

\* **@param** bariCase A BaRI user.

\*/

**public** Update(**final** BariCase bariCase) {

//Add a form.

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

//Handle required fields errors.

@Override

**protected** **void** onError() {

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

**if** (!title.checkRequired()) {

emptyFields.add("Overskrift");

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

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

}

**else** {

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

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

}

**if** (!type.checkRequired()) {

emptyFields.add("Type");

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

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

}

**else** {

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

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

}

**if** (!description.checkRequired()) {

emptyFields.add("Beskrivelse");

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

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

}

**else** {

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

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

}

StringBuilder errorMessage = **new** StringBuilder();

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

errorMessage.append("Feltet ");

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

.append("'");

}

**else** {

errorMessage.append("Felterne ");

**int** fieldCounter = 1;

**for** (String field : emptyFields) {

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

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

errorMessage.append(", ");

}

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

errorMessage.append(" og ");

}

fieldCounter++;

}

}

errorMessage.append(" skal udfyldes.");

setErrorMessage(errorMessage.toString());

}

};

add(form);

//Add form fields.

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

title.setRequired(**true**);

form.add(title);

type = **new** DropDownChoice("type",

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

Type.getDescriptions());

type.setRequired(**true**);

form.add(type);

form.add(**new** TextField("user", **new** Model(bariCase.getBariUser())));

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

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

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

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

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

caseStatus = **new** DropDownChoice("caseStatus",

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

CaseStatus.getDescriptions());

caseStatus.setRequired(**true**);

form.add(caseStatus);

devStatus = **new** DropDownChoice("devStatus",

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

DevStatus.getDescriptions());

devStatus.setRequired(**true**);

form.add(devStatus);

description = **new** TextArea("description",

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

description.setRequired(**true**);

form.add(description);

conclusion = **new** TextArea("conclusion",

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

form.add(conclusion);

//Add form links and buttons

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

@Override

**public** **void** onClick() {

Page page = **new** Discussion(bariCase);

setResponsePage(page);

}

});

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

@Override

**public** **void** onClick() {

setResponsePage(Overview.**class**);

}

});

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

@Override

**public** **void** onSubmit() {

bariCase.setTitle(title.getModelObject());

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

bariCase.setCaseStatus(

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

bariCase.setDevStatus(

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

bariCase.setDescription(description.getModelObject());

bariCase.setConclusion(conclusion.getModelObject());

**try** {

BariCaseBusiness.update(bariCase);

setResponsePage(Overview.**class**);

}

**catch** (StaleObjectStateException sose) {

setErrorMessage("Sagen kan ikke gemmes, " +

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

}

}

};

form.add(saveButton);

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

@Override

**public** **void** onClick() {

BariCaseBusiness.delete(bariCase);

setResponsePage(Overview.**class**);

}

};

deleteLink.add(**new** JS\_EventConfirmation("onclick", "Er du sikker pÃ¥" +

" at du vil slette?"));

form.add(deleteLink);

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

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

title.setEnabled(**false**);

type.setEnabled(**false**);

caseStatus.setEnabled(**false**);

devStatus.setEnabled(**false**);

description.setEnabled(**false**);

conclusion.setEnabled(**false**);

saveButton.setEnabled(**false**);

}

}

/\*\*

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

\*/

**private** **class** JS\_EventConfirmation **extends** AttributeModifier {

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

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

}

@Override

**protected** String newValue(**final** String currentValue,

**final** String replacementValue) {

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

**if** (currentValue != **null**) {

result += currentValue + ";";

}

**return** result;

}

}

}

#### style.css

body {

margin:*0*; padding:*0*;

font-family:*times*;

}

div {

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

}

#container {

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

width:*800px*;

}

#header {

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

width:*100%*;

}

#content {

width:*100%*;

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

}

#error {

text-align: *center*;

color: *red*;

}

#main {

width:*100%*;

}

/\* Table layout \*/

table.pageablelistview {

margin-bottom: *10px*;

border-bottom: *1px* *solid*;

}

table.pageablelistview caption {

text-align: *left*;

}

table.pageablelistview tr {

padding-top: *2px*;

padding-bottom: *2px*;

}

table.pageablelistview trtable.pageablelistview caption {

text-align: *left*;

}.even {

background-color: *#ececec*;

}

table.pageablelistview tr.odd {

background-color: *#fff*;

}

table.pageablelistview tr td {

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

}

table.pageablelistview tr th {

color: *black*;

padding-top: *3px*;

padding-bottom: *3px*;

padding-left: *8px*;

padding-right: *30px*;

background-color: *#ececec*;

border-bottom: *1px* *solid*;

border-top: *1px* *solid*;

text-align: *left*;

white-space: *nowrap*;

vertical-align: *middle*;

}

table.pageablelistview tr th {

background-position: *right*;

background-repeat:*no-repeat*;

}

table.pageablelistview tr th a {

font-weight: *normal*;

}

### Pakke dk.jsh.itdiplom.wsp.bari.bussiness

#### BariCaseBusiness.java

**package** dk.jsh.itdiplom.wsp.bari.bussiness;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariCase;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.CaseStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.Type;

**import** dk.jsh.itdiplom.wsp.bari.util.HibernateUtil;

**import** java.util.ArrayList;

**import** java.util.Date;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.hibernate.Session;

**import** org.hibernate.Transaction;

/\*\*

\* Business metods for BariCase.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** BariCaseBusiness {

/\*\*

\* Create new BariCase.

\*

\* **@param** bariCase BariCase

\*/

**public** **static** **void** saveNew(BariCase bariCase) {

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

Transaction tx = session.beginTransaction();

session.save(bariCase);

tx.commit();

session.close();

}

/\*\*

\* Get all BariCase objects.

\*

\* **@param** type type of BariCases to return.

\* **@param** caseStatus type of caseStatus to return.

\* **@return** a List of BariCase objects.

\*/

**public** **static** List<BariCase> getAllBariCases(Type type,

CaseStatus caseStatus) {

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

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

String hql = "select bariCase from "

+ "dk.jsh.itdiplom.wsp.bari.domain.BariCase bariCase "

+ "where bariCase.type = :type ";

**if** (caseStatus != **null**) {

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

}

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

Query query = session.createQuery(hql);

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

**if** (caseStatus != **null**) {

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

}

bariCases = query.list();

session.close();

**return** bariCases;

}

/\*\*

\* Update a BariCase.

\*

\* **@param** bariCase BariCase

\*/

**public** **static** **void** update(BariCase bariCase) {

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

**try** {

Transaction tx = session.beginTransaction();

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

bariCase.setFinished(**new** Date());

}

session.update(bariCase);

tx.commit();

}

**finally** {

session.close();

}

}

/\*\*

\* Delete a bariCase and all discussion messages.

\*

\* **@param** bariCase BariCase

\*/

**public** **static** **void** delete(BariCase bariCase) {

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

Transaction tx = session.beginTransaction();

String sql = "delete from DiscussionMessage where "

+ "bariCase\_id = :id";

Query query = session.createSQLQuery(sql);

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

query.executeUpdate();

session.delete(bariCase);

tx.commit();

session.close();

}

}

#### DiscussionMessageBusiness.java

**package** dk.jsh.itdiplom.wsp.bari.bussiness;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariCase;

**import** dk.jsh.itdiplom.wsp.bari.domain.DiscussionMessage;

**import** dk.jsh.itdiplom.wsp.bari.util.HibernateUtil;

**import** java.util.ArrayList;

**import** java.util.List;

**import** org.hibernate.Query;

**import** org.hibernate.Session;

**import** org.hibernate.Transaction;

/\*\*

\* Business metods for DescussionMessage.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** DiscussionMessageBusiness {

/\*\*

\* Create new DiscussionMessage.

\*

\* **@param** discussionMessage DiscussionMessage

\*/

**public** **static** **void** saveNew(DiscussionMessage discussionMessage) {

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

Transaction tx = session.beginTransaction();

session.save(discussionMessage);

tx.commit();

session.close();

}

/\*\*

\* Get all DiscussionMessages for a specific BariCase.

\*

\* **@param** bariCase BariCase.

\* **@return** a List of DiscussionMessage objects.

\*/

**public** **static** List<DiscussionMessage> getAllDiscussionMessages(

BariCase bariCase) {

List<DiscussionMessage> discussionMessages =

**new** ArrayList<DiscussionMessage>();

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

String hql = "select discussionMessage from "

+ "dk.jsh.itdiplom.wsp.bari.domain.DiscussionMessage discussionMessage “

+ "where bariCase.id = :id "

+ "order by discussionMessage.created";

Query query = session.createQuery(hql);

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

discussionMessages = query.list();

session.close();

**return** discussionMessages;

}

}

### Pakke dk.jsh.itdiplom.wsp.bari.domain

#### BariCase.java

**package** dk.jsh.itdiplom.wsp.bari.domain;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.CaseStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.DevStatus;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants.Type;

**import** java.io.Serializable;

**import** java.util.Date;

**import** javax.persistence.\*;

/\*\*

\* BariCase entity class.

\*

\* **@author** Jan S. Hansen

\*/

@Entity

**public** **class** BariCase **implements** Serializable {

**private** **static** **final** **long** serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

**protected** Long id;

@Version

**protected** Integer version;

**protected** String title;

@Enumerated(EnumType.STRING)

**protected** Type type;

**protected** String bariUser;

@Temporal(javax.persistence.TemporalType.TIMESTAMP)

**protected** Date created;

@Temporal(javax.persistence.TemporalType.TIMESTAMP)

**protected** Date finished;

@Enumerated(EnumType.STRING)

**protected** CaseStatus caseStatus;

@Enumerated(EnumType.STRING)

**protected** DevStatus devStatus;

**protected** String description;

**protected** String conclusion;

**public** BariCase() {

}

**public** BariCase(String title, Type type, String bariUser, Date created,

Date finished, CaseStatus caseStatus, DevStatus devStatus,

String description, String conclusion) {

**this**.title = title;

**this**.type = type;

**this**.bariUser = bariUser;

**this**.created = created;

**this**.finished = finished;

**this**.caseStatus = caseStatus;

**this**.devStatus = devStatus;

**this**.description = description;

**this**.conclusion = conclusion;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Integer getVersion() {

**return** version;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** Type getType() {

**return** type;

}

**public** **void** setType(Type type) {

**this**.type = type;

}

**public** String getBariUser() {

**return** bariUser;

}

**public** **void** setBariUser(String bariUser) {

**this**.bariUser = bariUser;

}

**public** Date getCreated() {

**return** created;

}

**public** **void** setCreated(Date created) {

**this**.created = created;

}

**public** Date getFinished() {

**return** finished;

}

**public** **void** setFinished(Date finished) {

**this**.finished = finished;

}

**public** CaseStatus getCaseStatus() {

**return** caseStatus;

}

**public** **void** setCaseStatus(CaseStatus caseStatus) {

**this**.caseStatus = caseStatus;

}

**public** DevStatus getDevStatus() {

**return** devStatus;

}

**public** **void** setDevStatus(DevStatus devStatus) {

**this**.devStatus = devStatus;

}

**public** String getDescription() {

**return** description;

}

**public** **void** setDescription(String description) {

**this**.description = description;

}

**public** String getConclusion() {

**return** conclusion;

}

**public** **void** setConclusion(String conclusion) {

**this**.conclusion = conclusion;

}

}

#### BariUser.java

**package** dk.jsh.itdiplom.wsp.bari.domain;

**import** java.io.Serializable;

/\*\*

\* BariUser entity class.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** BariUser **implements** Serializable {

**private** **static** **final** **long** serialVersionUID = 1L;

**private** String login;

**private** String password;

**private** String fullname;

**private** String email;

**public** BariUser() {

}

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

String email) {

**this**.login = login;

**this**.password = password;

**this**.fullname = fullname;

**this**.email = email;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getFullname() {

**return** fullname;

}

**public** **void** setFullname(String fullname) {

**this**.fullname = fullname;

}

**public** String getLogin() {

**return** login;

}

**public** **void** setLogin(String login) {

**this**.login = login;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

}

#### DiscussionMessage.java

**package** dk.jsh.itdiplom.wsp.bari.domain;

**import** java.io.Serializable;

**import** java.util.Date;

**import** javax.persistence.\*;

/\*\*

\* Discussion message entity class.

\*

\* **@author** Jan S. Hansen

\*/

@Entity

**public** **class** DiscussionMessage **implements** Serializable {

**private** **static** **final** **long** serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

**protected** Long id;

@Version

**protected** Integer version;

@ManyToOne

**protected** BariCase bariCase;

@Temporal(javax.persistence.TemporalType.TIMESTAMP)

**protected** Date created;

**protected** String bariUser;

**protected** String message;

**public** DiscussionMessage() {

}

**public** DiscussionMessage(BariCase bariCase, Date created, String bariUser,

String message) {

**this**.bariCase = bariCase;

**this**.created = created;

**this**.bariUser = bariUser;

**this**.message = message;

}

**public** String getBariUser() {

**return** bariUser;

}

**public** **void** setBariUser(String bariUser) {

**this**.bariUser = bariUser;

}

**public** Long getId() {

**return** id;

}

**public** **void** setId(Long id) {

**this**.id = id;

}

**public** Integer getVersion() {

**return** version;

}

**public** **void** setVersion(Integer version) {

**this**.version = version;

}

**public** BariCase getBariCase() {

**return** bariCase;

}

**public** **void** setBariCase(BariCase bariCase) {

**this**.bariCase = bariCase;

}

**public** Date getCreated() {

**return** created;

}

**public** **void** setCreated(Date created) {

**this**.created = created;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

}

#### Constants.java

**package** dk.jsh.itdiplom.wsp.bari.domain;

**import** java.util.ArrayList;

**import** java.util.List;

/\*\*

\* Constants and enums.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** Constants {

**private** Constants() {}

/\*\*

\* BaRI case type: REQUEST, ERROR.

\*/

**public** **enum** Type {

REQUEST("Nyt Ã¸nske"),

ERROR("Fejl");

**private** String description;

Type(String description) {

**this**.description = description;

}

**public** String getDescription() {

**return** description;

}

**public** **static** List<String> getDescriptions() {

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

descriptions.add(ERROR.description);

descriptions.add(REQUEST.description);

**return** descriptions;

}

**public** **static** Type getType(String description) {

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

**return** ERROR;

}

}

/\*\*

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

\*/

**public** **enum** CaseStatus {

NEW("Ny"),

CONSIDER("Behandles"),

APPROVED("Godkendt"),

REJECTED("Afvist"),

DONE("Afsluttet");

**private** String description;

CaseStatus(String description) {

**this**.description = description;

}

**public** String getDescription() {

**return** description;

}

**public** **static** List<String> getDescriptions() {

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

descriptions.add(NEW.description);

descriptions.add(CONSIDER.description);

descriptions.add(APPROVED.description);

descriptions.add(REJECTED.description);

descriptions.add(DONE.description);

**return** descriptions;

}

**public** **static** CaseStatus getCaseStatus(String description) {

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

**if** (CONSIDER.description.equals(description)) **return** CONSIDER;

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

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

**return** REJECTED;

}

}

/\*\*

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

\* INPRODUCTION.

\*/

**public** **enum** DevStatus {

NOTSTARTED("Ej begyndt"),

STARTED("I gang"),

READYTOTEST("Klar til test"),

TESTED("Testet"),

INPRODUCTION("I prod.");

**private** String description;

DevStatus(String description) {

**this**.description = description;

}

**public** String getDescription() {

**return** description;

}

**public** **static** List<String> getDescriptions() {

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

descriptions.add(NOTSTARTED.description);

descriptions.add(STARTED.description);

descriptions.add(READYTOTEST.description);

descriptions.add(TESTED.description);

descriptions.add(INPRODUCTION.description);

**return** descriptions;

}

**public** **static** DevStatus getDevStatus(String description) {

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

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

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

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

**return** INPRODUCTION;

}

}

}

### Pakke dk.jsh.itdiplom.wsp.bari.util

#### HibernateUtil.java

**package** dk.jsh.itdiplom.wsp.bari.util;

**import** org.hibernate.\*;

**import** org.hibernate.cfg.\*;

/\*\*

\* Hibernate session factory.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** HibernateUtil {

**private** **static** SessionFactory sessionFactory;

**static** {

**try** {

sessionFactory =

**new** AnnotationConfiguration().configure().buildSessionFactory();

}

**catch** (Throwable ex) {

**throw** **new** ExceptionInInitializerError(ex);

}

}

/\*\*

\* Get a Hibernate session factory.

\*

\* **@return** a SessionFactory

\*/

**public** **static** SessionFactory getSessionFactory() {

**return** sessionFactory;

}

/\*\*

\* Close SessionFactory.

\*/

**public** **static** **void** shutdown() {

getSessionFactory().close();

}

}

### Pakke dk.jsh.itdiplom.wsp.bari.wicket.test

#### CreateNewTest.java

**package** dk.jsh.itdiplom.wsp.bari.wicket.test;

**import** dk.jsh.itdiplom.wsp.bari.domain.BariUser;

**import** dk.jsh.itdiplom.wsp.bari.domain.Constants;

**import** dk.jsh.itdiplom.wsp.bari.wicket.CreateNew;

**import** org.apache.wicket.util.tester.WicketTester;

**import** org.junit.Test;

**import** **static** org.junit.Assert.\*;

/\*\*

\* JUnit test for CreateNew page.

\*

\* **@author** Jan S. Hansen

\*/

**public** **class** CreateNewTest {

/\*\*

\* Test that required fields must be filled in.

\*/

@Test

**public** **void** TestRequiredFields() {

WicketTester wt = **new** WicketTester();

BariUser bariUser = **new** BariUser("login", "password", "Bruger navn",

"email");

CreateNew createNew = **new** CreateNew(bariUser);

wt.startPage(createNew);

wt.assertLabel("error", ""); //No errors yet

wt.setParameterForNextRequest("form:title", "");

wt.setParameterForNextRequest("form:user", "");

wt.setParameterForNextRequest("form:description", "");

wt.setParameterForNextRequest("form:type", Constants.Type.ERROR);

wt.submitForm("form");

String error = wt.getTagByWicketId("error").getValue();

assertTrue("Title/Overskrift should be part of the error message",

error.contains("Overskrift"));

assertTrue("User/Oprettet af should be part of the error message",

error.contains("Oprettet af"));

assertTrue("Descripten/Beskrivelse should be part of the error message",

error.contains("Beskrivelse"));

}

}

### SQL til dannelse af databasen

#### bari\_dll.sql

**alter** **table** DiscussionMessage

**drop** **constraint** FKBC64A29FF02FEB14;

**drop** **table** BariCase;

**drop** **table** DiscussionMessage;

**create** **table** BariCase (

id bigint **not** **null** generated always **as** identity,

bariUser **varchar**(255),

caseStatus **varchar**(255),

conclusion **varchar**(255),

created **timestamp**,

description **varchar**(255),

devStatus **varchar**(255),

finished **timestamp**,

title **varchar**(255),

**type** **varchar**(255),

version **integer**,

**primary** **key** (id)

);

**create** **table** DiscussionMessage (

id bigint **not** **null** generated always **as** identity,

bariUser **varchar**(255),

created **timestamp**,

message **varchar**(255),

version **integer**,

bariCase\_id bigint,

**primary** **key** (id)

);

**alter** **table** DiscussionMessage

**add** **constraint** FKBC64A29FF02FEB14

**foreign** **key** (bariCase\_id)

**references** BariCase;

### Konfigutationsfiler

#### Hibernate.cfg.xml

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

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

<hibernate-configuration>

<session-factory>

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

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

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

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

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

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

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

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

<!-- List of annotated classes -->

<mapping class=*"dk.jsh.itdiplom.wsp.bari.domain.BariCase"* />

<mapping class=*"dk.jsh.itdiplom.wsp.bari.domain.DiscussionMessage"* />

</session-factory>

</hibernate-configuration>

#### web.xml

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

<web-app version=*"2.5"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>

<filter>

<filter-name>WicketApplication</filter-name>

<filter-class>org.apache.wicket.protocol.http.WicketFilter</filter-class>

<init-param>

<param-name>applicationClassName</param-name>

<param-value>dk.jsh.itdiplom.wsp.bari.wicket.Application</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>WicketApplication</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<session-config><session-timeout>

30

</session-timeout></session-config>

<welcome-file-list>

<welcome-file/>

</welcome-file-list>

</web-app>

### Ant target til generering af databaseskema

Følgende er indsat i NetBeans build.xml (Projekt fil). Som er et Ant target der udfra hibernatekonfigfilen og de domaineklasser denne peger på, kan genere et databaseskema.

<taskdef name="hibernatetool"

classname="org.hibernate.tool.ant.HibernateToolTask"

classpathref="project.classpath"/>

<target name="schemaexport">

<hibernatetool destdir=".">

<classpath path="build/web/WEB-INF/classes" />

<annotationconfiguration configurationfile="src/java/hibernate.cfg.xml" />

<hbm2ddl

drop="true"

create="true"

export="true"

outputfilename="bari\_ddl.sql"

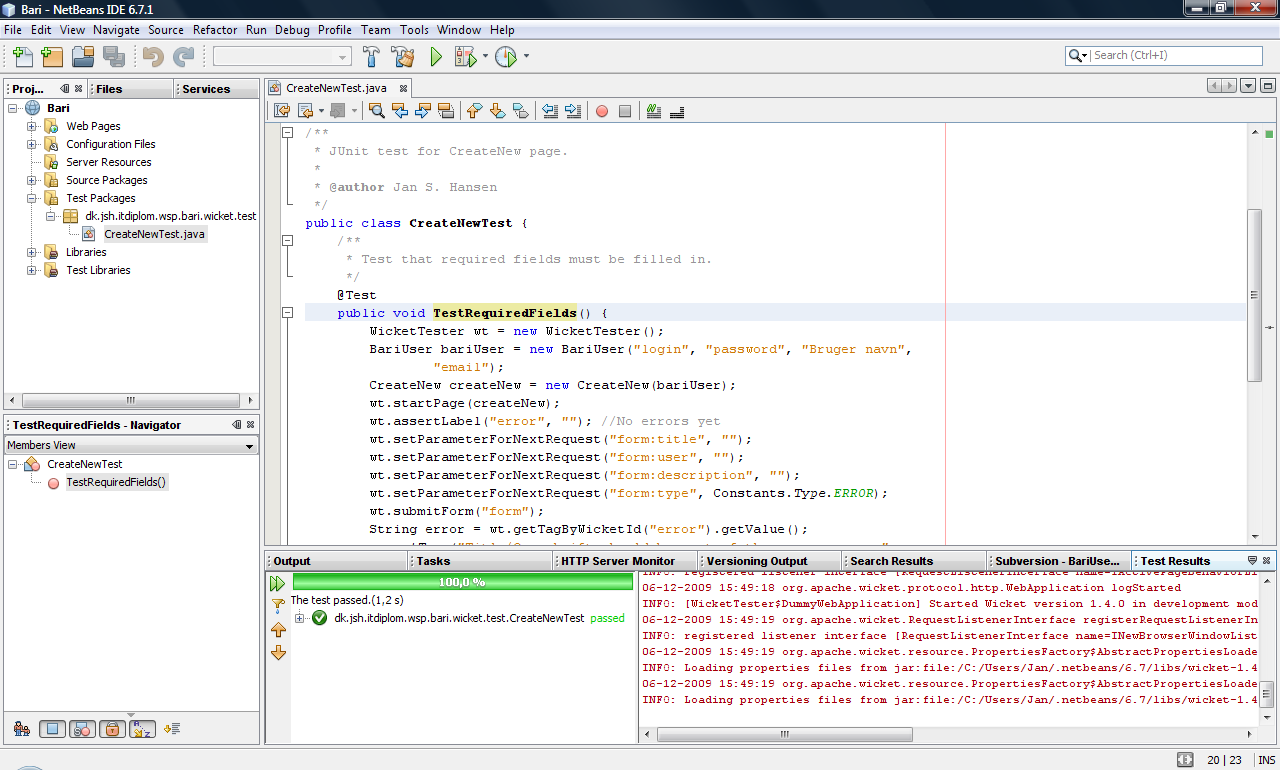
delimiter=";"

format="true" />

</hibernatetool>

</target>

## Eksempler på unit test



## Indhold på vedlagte CD

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

* Løsning – Indeholder java kode, htmlfiler m.m samt NetBeans projektfil.
* Program – Indeholder en bari.war samt JavaDB skemafil til oprettelse af databasen.
* Rapport – Indeholder denne rapport i Word 2007 format og diagrammer i Dia-format.

1. Wicket, se [wicket.apache.org](http://wicket.apache.org/). [↑](#footnote-ref-2)
2. Wicket in Action, se [www.manning.com/dashorst](http://www.manning.com/dashorst/) [↑](#footnote-ref-3)
3. Why Wicket, se [wicket.apache.org/introduction.html](http://wicket.apache.org/introduction.html) [↑](#footnote-ref-4)
4. Hibernate, se [www.hibernate.org](https://www.hibernate.org/) [↑](#footnote-ref-5)
5. Unit test, se [en.wikipedia.org/wiki/Unit\_testing](http://en.wikipedia.org/wiki/Unit_testing) [↑](#footnote-ref-6)
6. NetBeans IDE, se [netbeans.org](http://netbeans.org/) [↑](#footnote-ref-7)
7. JavaDB, se [developers.sun.com/javadb](http://developers.sun.com/javadb/) [↑](#footnote-ref-8)
8. Apache Tomcat, se [tomcat.apache.org](http://tomcat.apache.org/) [↑](#footnote-ref-9)
9. SubVersion, se [subversion.tigris.org](http://subversion.tigris.org/) [↑](#footnote-ref-10)
10. Dia, se [projects.gnome.org/dia](http://projects.gnome.org/dia/) [↑](#footnote-ref-11)
11. Wicket introduction, se [wicket.apache.org/introduction.html](http://wicket.apache.org/introduction.html) [↑](#footnote-ref-12)
12. Se OWASP Top 10, se [www.owasp.org/index.php/Top\_10\_2007](http://www.owasp.org/index.php/Top_10_2007) [↑](#footnote-ref-13)
13. HTTPS/SSL, se [en.wikipedia.org/wiki/Transport\_Layer\_Security](http://en.wikipedia.org/wiki/Transport_Layer_Security) [↑](#footnote-ref-14)
14. Man in the middle, se [en.wikipedia.org/wiki/Man-in-the-middle\_attack](http://en.wikipedia.org/wiki/Man-in-the-middle_attack) [↑](#footnote-ref-15)