PROJEKTOPGAVE FORÅR 2013
02324 VIDEREGÅENDE PROGRAMMERING.

CDIO DEL 4

AFLEVERINGSFRIST: MANDAG DEN 15/05 2013 KL. 05:00

Gruppe nr. 13

Denne rapport er afleveret via Campusnet (der skrives ikke under)

Denne rapport indeholder 11 sider incl. denne side.

s113730, Østergaard, Jesper 

 s123694, Celik ,Veli

s123849, Walbom, Oliver 

 s122966, Schmidt, Christopher

s123123, Kok, Mathias 

 s114211, Hviid, Nicklas

s123658, Bengtsson , Martine 

Videregående programmering - CDIO 4							
Time-regnskab							
Dato	**Deltager**	**Design**	**Impl.**	**Test**	**Dok.**	**Andet**	**Ialt**
28-04-2013	Jesper	1	1,5				2,5
-	Oliver	1	1,5				2,5
-	Martine	1	1,5				2,5
	Veli	1	1,5				2,5
	Mathias	1	1,5				2,5
	Christopher	1	1,5				2,5
06-05-2013							
	Jesper		2				2
	Oliver		2				2
	Martine		2				2
	Veli		2				2
	Mathias		2				2
	Christopher		2				2
	Nicklas		2				2
13-05-2013							
	Jesper			1	1		2
	Oliver			1	1		2
	Martine			1	1		2
	Veli			1	1		2
	Mathias			1	1		2
	Christopher			1	1		2
	Nicklas			1	1		2
14-05-2013							
	Jesper			1			1
	Oliver						0
	Martine						0
	Veli						0
	Christopher						0
	Nicklas			1			1
	Martine						0
	Sum	6	23	9	7	0	45

INDHOLD

FORORD

Denne rapport dokumenterer den sidste del ud af fire CDIO-delprojekter i faget 02324 Videregående programmering. Rapporten indeholder gruppens tanker og overvejelser om projektet, forklaringer af programmets virkemåde og domæne med tilhørende diagrammer (UML) og et testafsnit.

INDLEDNING

I denne opgave har vi igen lavet et web interface (vi lavede også et web-interface i CDIO delopgave 3). Vi har adgangskontrol i vores applikation i form af logind. Derfor har vi mulighed for at give administratorer flere valgmuligheder efter logind. Webapplikationen er i stand til at administrere (oprette, slette, rette, vise) operatøre af Mettler BBK vægten og skifte adgangskode. Vi har desuden en nettovægtsudregner applikation med i systemet. Naturligvis er det kun administratorer der har mulighed for at administrere og oprette operatøre.

Systemet er designet efter principperne i 3-lagsmodellen. Kontrollaget er implementeret som en HTTPServlet, datalaget er implementeret vha. container-klassen ArrayList, modellaget vha. JavaBeans og præsentationslaget som JSP-sider.

PROBLEMFORMULERING

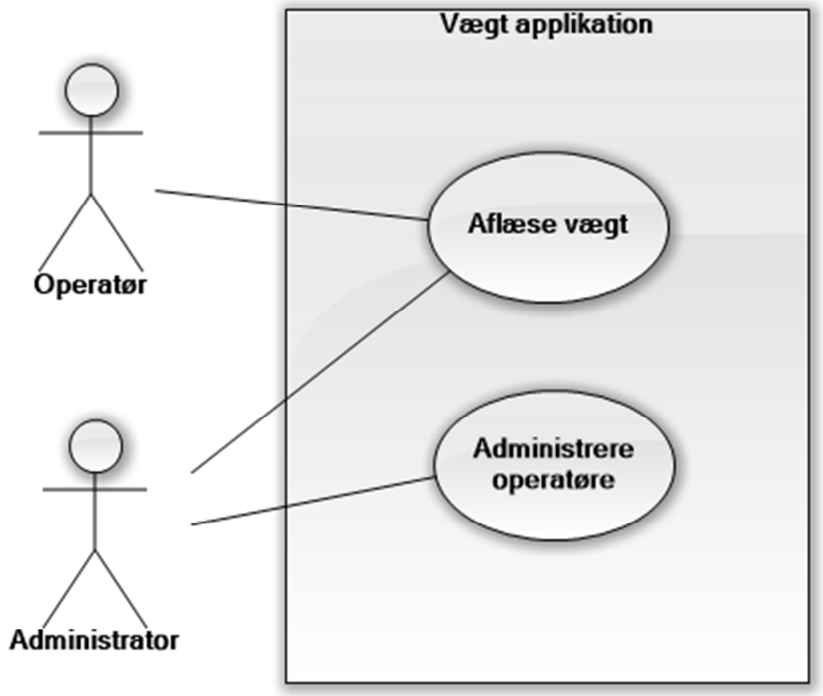
Til denne opgave skal der laves et web-interface. En operatør skal kunne logge ind med sit password og adgangskode, hvis koden er blevet godkendt vil man kunne ændre sin adgangskode. Hvis man er administrator i systemet kan man have alle mulighederne, som indebærer oprette, slette, rette og vise informationerne for de andre operatører. Alle oplysningerne skal kunne blive verificeret i systemet, og hvis de ikke overholder kravene, bliver der vist fejlmeddelelse, der informerer brugeren om hvordan

at de kan opnå kravene.
Vægten kan også tilgås fra dette web-interface.

KRAV

- Must:
    - WebInterface:
        - Et password beskyttet internet system, som kan benyttes af brugeren til at administere operatører.
    - Operatør-Oprettelse:
        - Udfyld efter forudbestemt standard (se Opgave beskrivelse for OperatoerDTO) og gem denne på Serveren.
    - Operatør-Sletning
        - Slet operatør ved operatørens id-nummer.
    - Operatør-Redigering
        - Rediger operatør data bortset fra Id.
    - Operatør-Visning
        - Vis operatører og deres data.
    - Operatør-password skift
        - Lad operatøre kunne skifte deres password.
    - Web-baseret test
        - Oprette en administartor der kan tilgå operatør administrationen, samt en simpel password beskyttet web-baseret vægt applikation.
    - Should Have:
        - ?
    - Could Have:
        - Samling af operatør visning og redigering
            - Hente data fra operatør-visning til automatisk udfyldning af felterne til redigering af operatøren via en dropdown menu.
    - Want to have:
        - Fejlbeskeder:
            - En opsummering af fejlindtastninger ved logind, Operatør sletning/redigering/oprettelse/password skift og vægt applikationen.
        - Successbeskeder:
            - En succesbesked når brugerens indtastninger er blevet valideret og handlingen er gennemført.

USECASE DIAGRAM

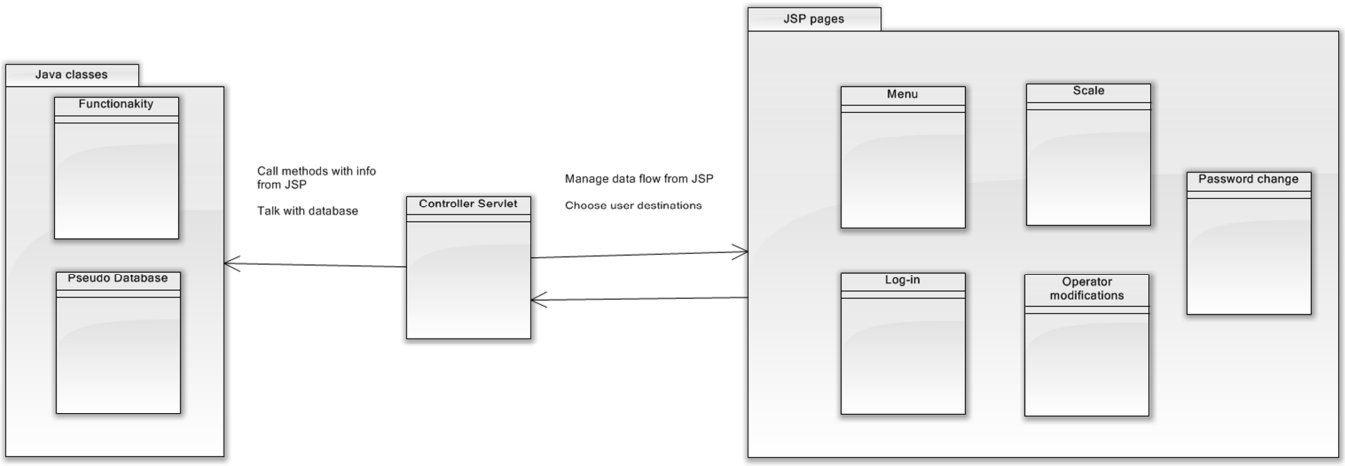


For at bruge vægt programmet er det første skridt altid at logge på, hver skal brugernavn og password indtastes, hvorefter programmet giver brugeren yderligere muligheder alt efter om brugeren er en operator eller en administrator. Så en precondition er at de er logget ind. Denne use case har vi valgt ikke at skrive på, da vi anser på den for at være for lille.

Når en bruger er logget på vil han/hun få stillet muligheder alt efter deres bruger type. En operatør har mulighed for at bruge test vægt delen samt ændre sit eget password. Grunden ti at vægt delen er med som en use case, er at den i længden vil blive en stor del af programmet, og du vil godt kunne lave afvejninger hele dagen. Dette kan ikke sige om at skifte password.

Alle administratorer har adgang til alle operatør funktioner. De har dog også adgang ”Administrere operatører” delen. Denne del giver muligheden for sletning af eksisterende brugere, oversigt over registrerede brugere, modificering at eksisterende brugere samt tilføjelse af nye brugere.

LØSNINGS DOMÆNE - MATHIAS



Løsningen til CDIO opgave nummer 4 er blevet lavet ud fra MVC modellen. Dette betyder at hele web-applikationen er bygget op omkring en controller servlet som styrer flowet af data mellem applikationens JSP sider samt kontrolere brugerens færdsel på disse. Når en bruger besøger applikationen vil controlleren bestemme hvilke side brugeren sendes til og alt efter brugerens valg sende ham/hende til andre sider, samt opdatere applikationens "database". Selve funktionaliteten og hukommelsen i applikationen ligger i nogle java klasser som tilgås af controlleren, dette betyder at en normal gennemgang af applikationen medfølger at først kaldes controlleren som sender brugeren til en JSP side, når brugeren så laver en handling på denne side sendes informationen så tilbage til controlleren som alt efter handling, henter funktionalitet fra java klasserne, skriver/læser i databasen og sender brugeren til en anden JSP side. Størstedelen af applikationens JSP sider og java klasser er modificerede udgaver af tidligere CDIO opgaver. Siderne der er i brug af denne applikation er beregnet til at: Logge ind, oprette og slette brugere, opdatere bruger password, bruge vægt applikation samt diverse menuer.

IMPLEMENTERING

Vores færdige resultat består af 8 Jsp-sider og 5 Java class-files. Programforløbet igennem disse filer er styret af den første side en bruger vil blive præsenteret for:

Efter MVC princippet, så er vores Controller klasse, servletten Run2.java den røde tråd i en brugers forløb igennem programmet. Denne klasse sørger for at sikrer sig at alle variabler er oprettet og initialiseret enten med application objektets data eller null-stillet. Hvis dette er første gang brugeren har kontakt til siden gør den som ovennævnt, opretter og nullstiller ellers tager den sig af at finde brugerens session data frem igen og initialisere alle variabler med de gemte data.

```java
public class Run2 extends HttpServlet {

    private static final long serialVersionUID = 1L;
    private BrugerValg valg = null;
    private IData2 d = null;  // interface reference til datalag
    private Login login = null;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public Run2() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        ServletContext application = request.getSession().getServletContext();
        HttpSession session = request.getSession();

        //tjekker om data er oprettet, ellers bliver det oprettet.
        d = (IData2) application.getAttribute("data");
        if(d==null)
        {
            d = new Data2();
            application.setAttribute("data", d);
        }

        //Opretter et nyt login objekt til sessionen hvis den ikke allerede findes
        login = (Login) session.getAttribute("login");
        if(login == null){
            login = new Login();
            login.setData(d);
            session.setAttribute("login", login);
        }

        //modtager oplysninger fra login siden
        String id;
        if((id = request.getParameter("id")) != null){
            login.setId(Integer.parseInt(id));
        }
        String pw;
        if((pw = request.getParameter("password")) != null){
            login.setAdgangskode(pw);
        }

        //Handling der skal udføres
        String handling = null;
        String[] params = request.getParameterValues("handling");
        if(params != null){
            handling = params[params.length-1];
        }

        //Tjekker om der er logget ind eller om der skal logges ud
        if ("log ind".equals(handling)) {
            login.tjekLogin();
        }
        if ("log_ud".equals(handling)) {
            login.setAdgangskode("");
        }

        //hvis man ikke er logget ind.
        if (!login.isLoggetInd()) {            // er brugeren logget korrekt ind?
            application.log("Bruger med "+login.getId()+" skal logge ind.");
            session.removeAttribute("valg");     // eller evt: session.invalidate()
            request.getRequestDispatcher("login.jsp?").forward(request,response);
            return;                              // afslut behandlingen af denne side
        }
```

Løbende over alle jsp siderne bliver Run2.java servletten også kaldt igennem html-formen's POST kommando, for at sørge for de respektive formfelter's indhold bliver kontrolleret og overført til session-objektets variabler.

Eksempelvis ved den første side en bruger bliver præsenteret for Login.jsp bliver brugeren præsenteret for tre html formular felter, henholdsvis ID, Password og sidst en knap "log ind". I jsp siden login.jsp er der i starten af html-formen oprettelse sat parameteren <form method="POST"> som indvirker at ved trykket på "log ind" knappen bliver Run2.java-servlettens doPost metode kaldt, som så igen kalder den samme klasses doGet metode som så løber alle session-objektets variabler igennem for at finde ud af den korrekte handling og viderestilling som brugeren har ønsket men også hvilken sider brugeren har ret til at se.

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<jsp:useBean id="user" class="data.OperatoerDTO2" type="data.OperatoerDTO2" scope="session"/>
<jsp:useBean id="login" class="funktionalitet.Login" type="funktionalitet.Login" scope="session"/>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login</title>
</head>
<body>

<h1>Log ind</h1>

<form method="POST">
<input type="hidden" name="handling" value="null">
<table>
<tr>
    <td>ID:</td>
    <td><input name="id" type="text"></td>
</tr>
<tr>
    <td>Adgangskode:</td>
    <td><input type="password" name="password"></td>
</tr>
<tr>
    <td></td>
    <td><input type="submit" name="handling" value="log ind"></td>
</tr>
</table>
</form>


</body>
</html>
```

Det næste brugeren bliver præsenteret for af Run2.java, er i direkte forbindelse med login.jsp naturligt nok en menu, nemlig menu.jsp. Her ses tydeligvis hvordan at brugerens før indtastede data afgøre hvad brugeren har ret til at se. Er brugerens session.attribute id ikke 10 jamen så bliver han/hun kun præsenteret for valgmulighederne "vægt applikation", "Skift Adgangskode", men derimod er id'et lig 10 så er brugeren identificeret som en administrator og for også valgmulighederne "Administrer Operatører" og "Opret Bruger". Hvilken af disse html-radio knapper brugeren vælger før at der bliver trykket på "Vælg menu punkt" og kaldt Run2.java's doPost metode afgøre hvad der bliver sat i session.objektets handling variable som igen bruges af Run2.java til at afgøre hvor brugere derefter skal dirigeres hen.

```jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<jsp:useBean id="user" class="data.OperatoerDTO2" type="data.OperatoerDTO2" scope="session"/>
<jsp:useBean id="login" class="funktionalitet.Login" type="funktionalitet.Login" scope="session"/>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Menu</title>
</head>
<body>
    <h1>Menu</h1>


<form method="POST">
    <input type="hidden" name="handling" value="null">
    <input type="radio" name="menuValg" value="weight">Vægt applikation<br>
    <input type="radio" name="menuValg" value="changePassword">Skift Adgangskode<br>

<%
    if(login.getId()==10)
    {
        %>
            <input type="radio" name="menuValg" value="showUsers">Administrer operatører<br>
            <input type="radio" name="menuValg" value="userForm">Opret bruger<br>
        <%
    }
%>
    <input type="submit" value="Vælg menu punkt">
</form>

<form method="POST">
    <input type="hidden" name="handling" value="log_ud">
    <input type="submit" value="Log ud">
</form>



</body>
</html>
```

En nævneværdig funktion i vores program er blandt andet vores evaluerings metoder. Disse metoder benytter sig af Regular Expressions til at validere brugerens input i de individuelle felter. Regular expressions har den fordel i brug at de er tæt på atomic i brug og er derved resource mæssigt ofte at foretrække frem for at itterere sig igennem en information's streng vha af loops. Dog skal det da nævnes at blandt mange regnes Regular Expressions i sig selv for at være en enorm fejl-kilde, nogen går så lang som at sige at "you had a problem you wanted to solve with Java, solved it with java and Regular Expressions. Now you just have two problems" Dette må nok tilskrives den ret svære gennemskuelighed der er i regular expressions. Blandt mange mener man at de ikke ligefrem fremmer vedligeholdelses graden af koden.

I vores tilfælde benytter vi os af Regular ekspressions (Regex) til at validere bruger oprettelse i administratør siden "Opret Bruger"-userForm.jsp, som navnet hentyder til er det vha. af denne sidde man opretter en bruger og overføre Html-formen's data til session objektet. Dog kun hvis alle Html-formens variabler klare sig igennem regex kontrollen, ellers bliver der udskrevet en fejlmeddelelse der beskriver hvilke karakterer det enkelte fejlramte felt skal indeholde for at kunne valideres.

```
private boolean checkRegex(String REGEX, String INPUT) {
    Pattern pattern = Pattern.compile(REGEX);
    Matcher matcher = pattern.matcher(INPUT);

    return matcher.matches();
}

/** Operat�r navn min. 2 max.. 20 karakterer */
public boolean checkName(String navn)
{
    String REGEX = "^[a-zA-Z[\\s]]{2,20}$+";
    return checkRegex(REGEX,navn);
}

/** Operat�r initialer min. 2 max. 3 karakterer */
public boolean checkIni(String ini)
{
    String REGEX ="^[a-zA-Z[\\s]]{2,3}$+";
    return checkRegex(REGEX, ini);
}

/** Operat�r cpr-nr 10 karakterer */
public boolean checkCpr(String cpr)
{
    String REGEX = "^[0-9]{10,10}$+";
    return checkRegex(REGEX, cpr);
}
/** Operat�r password min. 7 max. 8 karakterer */
public boolean checkPassword(String password)
{
    String REGEX = "^[0-9[a-zA-Z[\\-\\.\\+\\?[_!=[\\s]]]]]{7,8}$+";
    return checkRegex(REGEX, password);
}
```

TEST

Login fejl:

Den første fejl vi fandt var når der indtastes et tegn, bogstav, mellemrum, kommatal, meget høj int (99999999999999999999999999999999999999999999999) eller ingenting i id feltet i login, fås NumberFormatException.

Vægt applikation fejl:

Den næste fejl, der blev fundet var, at der under indtastning af fejlmeddelelsen i vægt applikationen var blevet tastet forkert, så der var byttet om på tarra og brutto så: "Tarra skal være større end brutto", burde være skrevet som: "Brutto skal være større end tarra".
I selve vægt applikationen var det også muligt at bruge en negativ tarra og brutto og få en positiv netto vægt på følgende måde:

Tarra: -8.0 Brutto: -2.0 Netto vaegt er: 6.0.

Derud over blev det opdaget at "<" (og andre tegn) kunne bruges i beregninger og give mærkelige resultater som i disse eksempler:

Tarra: <<<<<<<<<<<<<<<<<<<<< Brutto: 999999999999999999999

Netto vaegt er: 1.0E35

og

Tarra: < Brutto: 9

Netto vaegt er: 9

Ved de input hvor både tarra og brutto udelukkende bestod af tegn, blev netto lig 0.

Eksemplerne kunne tyde på, at tegn som input blev fortolket som 0'er af programmet.

Vægtapplicationen viste heller ingen fejlbesked, når et tal og bogstav blev kombineret. Fx:

Tarra: 1s Brutto: 1d

Netto vaegt er: 1.0

Tarra: 1f Brutto: 1h

"Tarra skal være større end brutto"

De fleste tal + bogstav kombinationer gav beskeden: "Netto vaegt er: 0.0".

Øvrige fejl:

Alle sider pånær changePw.jsp og chooseUser.jsp kan tilgås direkte fra url'en ved at skrive sidens adresse. Dog kan siderne ikke bruges til noget videre. (Der sker ikke noget, når man trykker på knapperne.) Det burde dog slet ikke være muligt for brugeren at tilgå siderne udenom kontrolleren. Forsøger man at tilgå chooseUser.jsp via url'en fås en NullpointerException.

Udover det lever password tjekket ikke op til DTU's standard. Umiddelbart vil enhver række af 7-8 karakterer accepteres, heriblandt "1234567", "aaaaaaaa" og "-------".

Resten af de testede funktioner virkede, som de skulle. Det var ikke muligt at slette sig selv. Administratoren kunne slette/redigere alle andre brugere. Man kunne kun logge ind med et brugernavn og en dertil hørende adgangskode som fandtes i systemet.

Afsluttende kommentar / konklusion

Vi har fået lavet et web-interface så det er muligt at logge ind for derefter at blive sendt videre til menuen. Hvis der logges ind som administrator vil der blive vist lidt flere menupunkter. Programmets flow styres gennem en servlet-controller, som bestemmer hvilken side der skal vises.