Projektopgave forår 2013

02324 Videregående programering.

CDIO del 3

Afleveringsfrist: Mandag den 29/04 2013 Kl. 05:00

Gruppe nr. 13

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

Denne rapport indeholder 10 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 3							
Time-regnskab							
Dato	**Deltager**	**Design**	**Impl.**	**Test**	**Dok.**	**Andet**	**Ialt**
08-04-2013	Jesper	1				1	2
-	Oliver	1				1	2
-	Martine	1				1	2
	Veli	1				1	2
	Mathias	1				1	2
	Chrisopher	1				1	2
15-04-2013							
	Jesper		2				2
	Oliver		2				2
	Martine		2				2
	Veli		2				2
	Mathias		2				2
	Chrisopher		2				2
	Nicklas		2				2
22-04-2013							
	Jesper			1	1		2
	Oliver				2		2
	Martine		2				2
	Veli				2		2
	Mathias				2		2
	Chrisopher				2		2
	Nicklas				2		2
27-04-2013							
	Jesper				1		1
	Oliver				1		1
	Martine				1		1
	Veli				1		1
	Chrisopher				1		1
	Nicklas				1		1
	Martine				1		1
28-04-2013							
	Jesper				1		1
	Oliver				1		1
	Martine				1		1
	Veli				1		1
	Chrisopher				1		1
	Nicklas				1		1
	Martine				1		1
	Sum	6	16	1	25	6	54

Indhold

Forord

Dette er del 3 af CDIO projektet, i kursus 02324, lavet af gruppe 13. Denne rapport vil forklare vores projekt, beskrive programmet lavet i opgaven, samt vise UML og forklarende tekst til projektet.

Indledning

CDIO opgave del 3 omhandler et web-interface, der giver mulighed for at oprette en ny recept i forbindelse med brug af vægten, hvor den nye recept laves til batches, som vægten skal afveje. Web interfacet skal validere de input der kommer fra brugeren, så det ikke er muligt at indtaste ugyldig data.

Problemformulering

Til denne opgave skal der laves et web-interface. En bruger skal kunne oprette nye recepter via web-interfacet. Når der modtages data fra brugeren til en nyoprettet recept, skal web-interfacet verificere om dette input er korrekt og enten give brugeren en success eller fejl besked. Hvis brugerens input er blevet valideret så skal denne recept gemmes til senere brug.

Krav

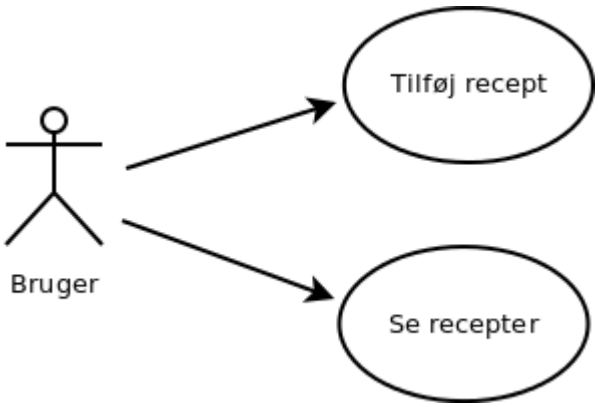
Krav til Del 3. indeholder ikke det store analyse arbejde, så er de dog essentielle for det videre projekt som de tydeligvis lægger op til.

Jeg har bedømt de nævnte krav efter MoSCoW metoden:

- Must:
    - WebInterface:
        - Et internet system, som kan benyttes af brugeren til at oprette recepter(se Recept Redigering).

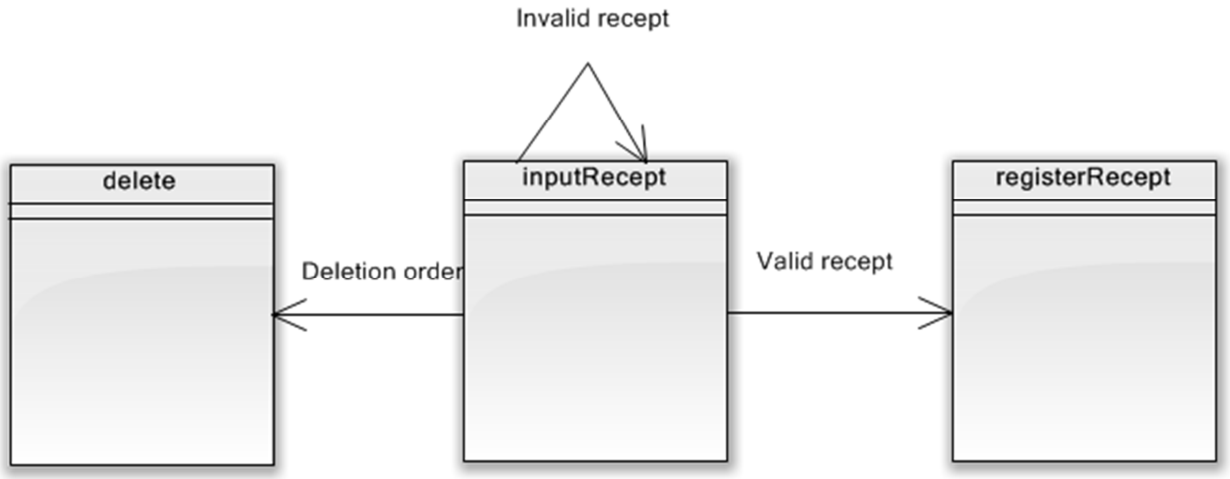
    - Recept-Oprettelse:
        - Udfyld efter forudbestemte standard (se Opgave beskrivelse for data formularen) og gem denne på Serveren.
- Should Have:
    - Batch produktion af Tidligere oprettet Recepter:
        - Hente tidligere oprettet recepter fra serveren og oprette x...xn batches af recepten.
- Could Have:
    - Webside til Batch-formular:
        - En formular til at indtaste data'en i.
- Want to have:
    - Webside til fejlbeskeder:
        - En opsummering af mulige fejlbeskeder fra tidligere handlinger.
    - Webside til resultat efter validering:
        - Gennemgang af Recepten.
    - Recept redigering:
        - Såvel som oprettelse af en recept så ligeledes mulighed for at redigere receptens indhold, eller helt slette den.

Usecase diagram



Vores hjemmeside har én aktør, brugeren. Denne aktør kan enten tilføje en ny recept eller se alle recepter. Brugeren tilføjer et nyt recept ved at skrive de nødvendige informationer på jsp siden inputRecept.jsp. Kun hvis informationerne er gyldige, dirigeres brugeren videre til jsp siden registerRecept. Herfra gemmes recepten i en fil og alle recepter vises. Når "Se recepter" er en use case for sig selv, er det fordi brugeren også kan gå direkte til registerRecept.jsp og se alle recepter uden samtidig at skulle tilføje et nyt recept.

Løsnings domæne



Opgavebeskrivelsen til denne opgave forslår at der skal laves 3 jsp sider. En til at skrive data ind, en til at vise at dataen er valid og en til hvis den ikke er, og vise hvad der problemet er. Designet brugt her er dog lidt anderledes bygget op, der er stadig tre sider, men deres funktion er anderledes. Der er en hovedside kaldet inputRecept, denne sides bruges til at indsende nye recepter, og i fald af en indvalid recept bruges den også som et error site, der viser hvor valideringen fejler. Den anden side, registerRecept bruges til at oprette recepten i databasen, i fald at recepten er valid (i denne version kun en pseudo-database). Den sidste side kaldet delete bruges til at slette recepter fra databasen.

Implementering

Vores løsning består af 3 jsp-filer; inputRecept, registerRecept og delete. inputRecept indeholder fomularen hvor man indtaster recepten, konvertering af formularinput og kontrol af inputtet.

Et udsnit af receptformularen ses herunder. Formularen er en normal html form, der anvender metoden GET. Dette betyder at formularens indhold videregives til andre sider igennem URL'en. I dette tilfælde videregives formularens indhold til siden selv (inputRecept.jsp), da det er her kontrol af inputtet foregår.

Såfremt at der er blevet trykket på knappen "Registrer" (submit knappen) og recepten ikke er valid (hvis der er fejl i indtastningerne) udskrives der oven over hvert input felt en fejlbesked. Denne fejlbesked bliver sat lig en tom streng for felter hvor inputtet i feltet er valid, således at brugeren ikke ser en fejlbesked. Noget andet som skal bemærkes er der som value for alle felter udskrives en streng fra programmet.

```
<form method = "GET" action = "inputRecept.jsp">
    <%
    if(submitted != null)
    {
        out.println(receiptError);
    }
    %>
    Receipt nr: <input type = "text" name = "receptNo" value = "<%= receiptNo %>"><br />
    <%
    if(submitted != null)
    {
        out.println(productMatchError);
    }
    %>
    Vare nr: <input type = "text" name = "productNo" value = "<%= productNo %>"><br />
    Vare navn: <input type = "text" name = "productName" value = "<%= productName %>"><br />
    .........
    <input type = "submit" name = "submit" value = "Registrer">
</form>
```

Disse værdistrenge initialiseres i starten af siden til at være lig null. Desuden oprettes strengvariabler til at indeholde fejlbeskederne og recepten sættes til at være valid.

```
<%
String receiptNo = null;
String productNo = null;
String productName = null;
String desiredWeight = null;
String tolerance = null;

String receiptError;
String productMatchError;
String weightError;
String toleranceError;

boolean validReceipt = true;
%>
```

I starten af sidens body sektion (oven over formularen) hentes værdien af submit. Denne bruges til at tjekke hvorvidt der er blevet trykket på “Registrer”-knappen, altså hvorvidt siden er blevet genindlæst efter at der blevet indtastet i formularen. Hvis siden er blevet genindlæst hentes værdierne af alle felter og disse gemmes i de tilhørende strenge. Herefter kontrolleres alle input vha. tilhørende kontrolmetoder. Disse metoder returnerer en fejlmeddelelse, som gemmes i en streng. Metoderne returnerer en tom streng hvis inputtet er valid, ellers returneres en fejlmeddelelse. Herefter tjekkes for alle fejlstrenge, hvorvidt strengen er forskellig fra tom (hvis inputtet ikke var valid). Hvis inputtet ikke var valid sættes det pågældende felts værdistreng til en tom streng. Dette medfører at alle ikke valide indtastninger i formularen fjernes. Samtidig sættes recepten til ikke at være valid. Hvis recepten er valid (hvis der ikke har været fejl i indtastningerne) gemmes alle felternes indtastninger i session objektet og browseren videresendes til siden registerRecept.jsp.

```
    <body>

<%
String submitted = request.getParameter("submit");

if(submitted != null)
{
    receiptNo = request.getParameter("receptNo");
    productNo = request.getParameter("productNo");
    productName = request.getParameter("productName");
    desiredWeight = request.getParameter("desiredWeight");
    tolerance = request.getParameter("tolerance");
}

receiptError = validReceiptNo(toInt(receiptNo));
productMatchError = validProductMach(productNo, productName);
weightError = validWeight(toDouble(desiredWeight));
toleranceError = validTolerance(toDouble(tolerance), toDouble(desiredWeight));

if(receiptError != "")
{
    receiptNo = "";
    validReceipt = false;
}
if(productMatchError != "")
{
    productNo = "";
    productName = "";
    validReceipt = false;
}
.............
if(validReceipt == true)
{
    session.setAttribute("receptNo", receiptNo);
    session.setAttribute("productNo", productNo);
    session.setAttribute("productName", productName);
    session.setAttribute("desiredWeight", desiredWeight);
    session.setAttribute("tolerance", tolerance);

    response.sendRedirect("registerRecept.jsp");
}

%>
```

Siden registerRecept.jsp opretter en fil med navn recepts.txt på serverens harddisk hvis filen ikke allerede eksisterer. Herefter tjekkes hvorvidt der er blevet lagt en attribut op i sessionsobjektet (hvis der er blevet indtastet en valid recept). Hvis der er det hentes alle disse attributer (alle felters indtastninger) og gemmes i strenge. Herefter oprettes et FileWriter objekt på vores fil. Desuden oprettes en BufferedWriter til FileWriter-objektet. Nederst i tekstfilen skrives formularens indtastninger i CSV-format. Desuden udskrives stien til recept.txt og sessionsobjektet, anvendt til at overføre indtastningerne, lukkes ned.

```
File receptFile = new File(application.getRealPath("/recepts.txt"));
try
{
    receptFile.createNewFile();

    // Tjekker, om et nyt recept skal gemmes og gemmer det
    if(session.getAttribute("receptNo") != null)
    {
        String receiptNo = (String)session.getAttribute("receptNo");
        String productNo = (String)session.getAttribute("productNo");
        String productName = (String)session.getAttribute("productName");
        String desiredWeight = (String)session.getAttribute("desiredWeight");
        String tolerance = (String)session.getAttribute("tolerance");

        FileWriter fileWriter = new FileWriter(receptFile, true);
        BufferedWriter outStream = new BufferedWriter(fileWriter);
        outStream.write(receiptNo + ";" + productNo + ";" + productName + ";"
        outStream.close();
        fileWriter.close();

        out.println("Din recept er nu gemt i filen:");
        out.println(receptFile.getAbsoluteFile() + "<br/><br/>");
        session.invalidate();
    }
```

Herefter udskrives alle recepter i filen. Først oprettes de nødvendige streamhåndteringsobjekter, en ArrayList recept til at indeholde alle recepterne og en streng recept til at indeholde de enkelte undervejs, som de indlæses. Linjerne i filen indlæses en efter en indtil bunden af filen nås. Hvis der overhovedet er recepter (linjer) i filen udskrives disse med en nydelig HTML-formatering, ellers udskrives at der ingen recepter er registreret.

```
// Hent alle recepter fra recepts.txt
FileInputStream fstream = new FileInputStream(receptFile);
DataInputStream in = new DataInputStream(fstream);
BufferedReader br = new BufferedReader(new InputStreamReader(in));
ArrayList<String[]> recepts = new ArrayList<String[]>();
String recept;
while((recept = br.readLine()) != null)
    recepts.add(recept.split(";"));

// Vis tabel over recepter, hvis der er nogle
if(recepts.size() != 0){
    String[] categories = {"Receptnummer", "Produktnummer", "Produktnavn",
            "Desired weight", "Tolerance"};
    out.print("<h2>Tabel over recepter</h2>");
    for(int i = 0; i < recepts.size(); i++)
    {
        out.print("<p>");
        for(int j = 0; j < categories.length; j++)
            out.print(categories[j] + ": " + recepts.get(i)[j] + "<br/>");
        out.print("</p>");
    }
}
else
    out.print("<p>Der er i øjeblikket ingen recepter registreret.</p>");

br.close();
in.close();
```

Test

Det første vi startede at teste hvad der skete hvis man ikke indtastede noget i nogen af felterne. Skærmbilledet herunder viser resultatet, hvor vi kan se at alle felter kommer med en sigende fejlbesked.

Det næste der skulle testes var om det var muligt at indtaste bogstaver i stedet for tal, i de felter der krævede tal som input, dvs. alle andre felter end produktnavnet.

Her fik vi følgende resultat:

Som det ses, så kunne programmet håndtere dette, ved hjælp af exceptions håndtering.

Ellers skulle vi have testet om programmet tjekkede de rigtige intervaller, f.eks. at recept nr er indenfor intervallet 1-99999999. Her valgte vi helt naturligt at tjekke om programmet kunne registrere tallene 1 og 99999999 da de er endepunkterne for det gyldige interval. Udover dette skulle der selvfølgelig også tjekkes om programmet kunne registrere tallene 0 og 100000000, da det er de to tal der er lige udenfor det gyldige interval. Samme fremgangsmåde brugte vi til de 5 andre inputs.

Som det kan ses på disse skærmbilleder, gik første del af testen fint, og det bliver registreret at de 2 tal var udenfor det gyldige interval.

Næste del var at teste om de 2 endepunkter i intervallet blev oprettet, korrekt. Dette gjorde de, så vi kan konkludere at vi har fået afgrænset intervallerne korrekt i vores program.

Den sidste del der skulle testes var om programmet slettede de rigtige recepter. Vores resultat viste at programmet slettede den rigtige recept, og gemte resten til tekstfilen igen.

Afsluttende kommentar / konklusion

Vi har fået oprettet en web-interface hvor det er muligt at oprette en ny recept. Web-interfacet tjekker om alle inputs er gyldige. Hvis de ikke er det bliver der udskrevet en sigende fejlbesked til brugeren. Efter brugeren har indtastet gyldige værdier for recepten, bliver brugeren sendt videre til en side der giver et overblik over alle recepter. Disse er gemt i en .txt på computeren. Brugeren har mulighed for at slette en recept hvis dette ønskes.