Danmarks Tekniske Universitet

Lokalereservation På DTU

Bachelorprojekt

Patrick Schøller Larsen s113415 og Frederik Skjoldborg s113430
15-06-2015

Resume

Skemalægning på DTU sker som det er nu manuelt og derfor er det interessant at se på om man kan gøre noget for at gøre dette nemmere for dem der lægge skemaer.

Vi har med dette bachelorprojekt forsøgt at hjælpe dem, ved at gå ud fra de samme filer som administrationen bruger nu, og skabe et program der kan opstille dette på en overskuelig måde.

Til dette har vi brugt GWT (Google Web Toolkit) og en MySQL-database.

Dette er sket gennem forskellige overvejelser af måder at løse de delproblemer vi er stødt på undervejs.

Ud fra de overvejelser vi har foretaget er der skabt en prototype til hvordan problemet kan blive mere overskueligt, både for administrationen og for de daglige brugere af systemet.

Abstract

Scheduling at DTU happens as it is right now very manually and therefore it is interesting to look at whether it is possible to do something to make this easier for them to make the schemas.

At this Bachelor project, we have tried to help them, by going from the same files as the administration uses, and create a program that can set this in a clear way.

For this, we used GWT (Google Web Toolkit) and a MYSQL database.

This has happened through different considerations of ways to solve the sub-problems we have encountered throughout this project.

Based on the considerations we have created a prototype for how the problem can be more manageable, both for administrators and for the daily users of the system.

Forord

Denne opgave er den afsluttende del af bacheloruddannelsen i Softwareteknologi på Danmarks Tekniske Universitet.
Projektet er udarbejdet på Institut for Matematik og Computer Science under vejledning af Stig Høgh, lektor ved DTU Compute. Projektet tæller 15 ETCS point og har løbet fra 2. februar 2015 til 15.juni 2015.

Denne rapport er resultatet af bachelorprojektet "Lokalereservation på DTU" og er udarbejdet af Patrick Schøller Larsen, s113415 og Frederik Skjoldborg, s113430. Rapporten er blevet til undervejs i projektets forløb og er en sammenfatning af hele udviklingsarbejdet.
Udviklingen af programmet samt udarbejdningen af denne rapport er foretaget af undertegnede og arbejdsbyrden er fordelt ligeligt.

Vi vil gerne takke vores vejleder Stig Høgh, for god vejledning og et spændende projekt.

X _____

Frederik Skjoldborg, s113430

X _____

Patrick Schøller Larsen, s113415

Indhold

Indledning

Før hvert semester på DTU lægger lærerne ønsker til undervisningslokaler ligesom de studerende gør det med kurser, gennem en funktion på campusnet der gemmer disse ønsker i en Excel fil, og det er på baggrund af denne fil der bliver lagt skemaer.
Der er ikke taget højde for dobbeltbookninger i filen, så disse kommer først frem, som skemaet bliver lagt. Der er mulighed for at skrive en kommentar til sit ønske, som især bliver relevant i tilfælde af dobbeltbookninger.
Som det fungerer nu lægges skemaerne manuelt ud fra Excel filen, der bliver udskrevet fra campusnet. Denne metode er tidskrævende, bla. fordi der skal bruges en del tid på at sikre der ikke sker dobbeltbookninger.
Formålet med dette projekt er at kigge på en alternativ måde at løse skemalægningen på ved hjælp af et computerprogram.
Da noget af det der tager lang tid er at sikre man ikke har dobbeltbookninger, har en vigtig del af projektet gået på at skabe en visning der giver et nemt og overskueligt overblik over dobbeltbookninger.
Det har vi gjort ved hjælp af en MySQL-database og GWT(Google Web Toolkit).

Det første afsnit omhandler hvad GWT er, hvordan vi har brugt det og hvorfor vi har brugt det.
Derefter kommer et afsnit om den måde vi har opsat vores database og hvordan vi forbinder den med resten af programmet.
Vi har valgt at forklare hvordan programmet fungerer, ved at tage de ting vi anser for de vigtigste og så gennemgå dem en for en. Dette sker via en analyse af hvorfor de ting vi har valgt er vigtige, en diskussion af forskellige måder problemet kunne være løst på, og til sidst hvad vi har valgt og hvorfor vi har valgt det. Her kommer vi også ind på mulighed for videreudvikling.
Så kommer en lidt mere teknisk gennemgang af hvordan programmet fungerer, og til sidst konkluderer vi på projektet.

Hvad er GWT

GWT eller Google Web Toolkit som det står for, er et open source værktøj som giver web udviklere en måde at lave og vedligeholde komplicerede Javascript front end applikationer i Java kode ved hjælp af deres fortrukne Java udviklingsværktøj.

Ved brug af GWT er det muligt at opbygge en hel web applikation fra bunden eller lave enkelte knapper eller andre funktioner som kan implementeres i allerede oprettede web applikationer.

GWT gør samarbejdet mellem web-siden og server-siden meget simplere ved hjælp af GWT RPC (Remote Procedure Call) som gør det nemt at udveksle Java objekter mellem klient-siden og server-siden over http. Den server-side kode der bliver brugt på klient-siden bliver kaldt en service.

Implementereingen af GWT RPC service er baseret på kendt Java servlet arkitektur.

GWT RPC sker ved hjælp af asynkrone kald, så hvis man vil kunne kalde en database f.eks MySql skal man sikre sig at man har slået Google App Engine fra da det kan blokere de asynkrone kald der skal til for at kontakte databasen.

Grunden til at det er smart med asynkrone kald som RPC er fordi kalderen ikke venter på svar og dermed stopper hele applikationen indtil den får det svar den skal bruge, men det sker sideløbende med resten af de ting der sker på siden.
Fordi man ved brug af RPC forventer at få svar, er det vigtigt at fortælle applikationen hvad der skal ske når kaldet lykkes, men også hvad der skal ske hvis det fejler.

Før man begynder at arbejde med GWT skal man hente og installere Java SDK, og helst den nyeste version for at sikre sig at man har en version som er så optimal som mulig.

Derefter skal man vælge hvilket Java udviklingsværktøj man vil bruge. Vi valgte at bruge Eclipse, da det er det udviklingsværktøj vi har mest erfaring med, men man kan også bruge NetBeans eller et hvilket som helst andet Java udviklingsværktøj.

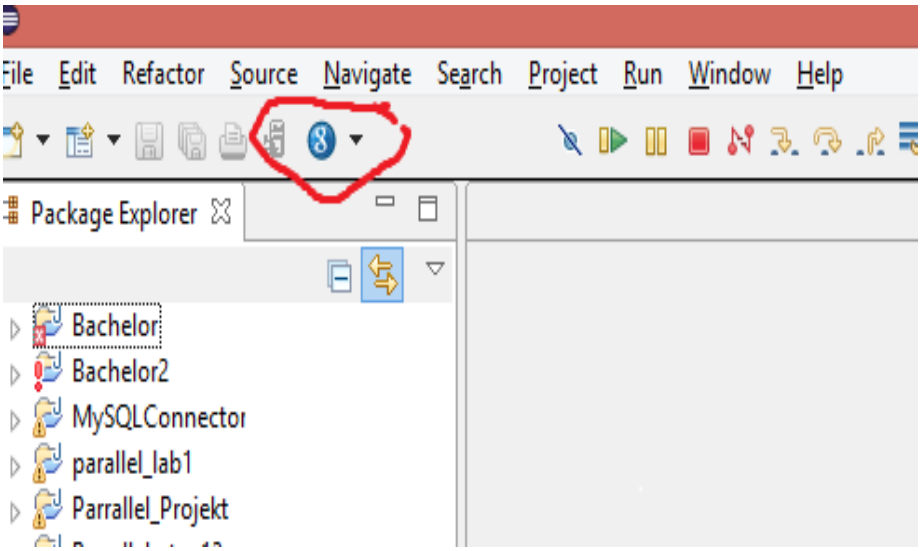
Lige meget hvilket Java udviklingsværktøj man vælger vil det stadig være de samme grundlæggende GWT koncepter.

Da vi har valgt at arbejde med Eclipse er det udviklingsværktøjet vi er gået ud fra i forhold til hvordan man gør tingene.

Næste trin er at installere Google Plugin for Eclipse og sørge for at den passer overens med den version af Eclipse man har installeret.

Derefter skal man hente Google Web Toolkit som for det meste kan downloades sammen med Google Plugin for Eclipse, men man skal være opmærksom på om versionen er den nyeste for ens styresystem og passer overens med den Java SDK version man har. Især hvis man gerne vil arbejde sammen med andre er det vigtigt at have de samme versioner.

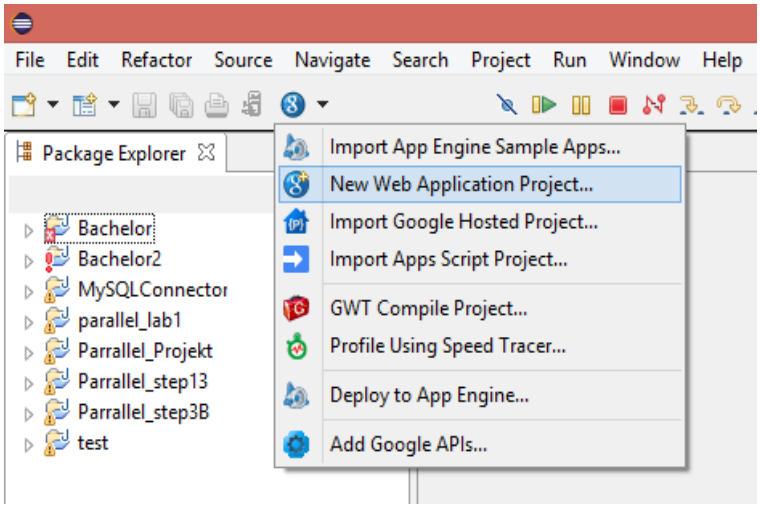
Når man har installeret Google Plugin til Eclipse vil man se et blåt logo oppe i værktøjslinjen. Logoet er vist på figur 1.



Figur 1 google plugin

Vil man starte et nyt GWT projekt trykker man på GWT logoet og vælger ”New Web Application Project”
Som vist på figur 2.

Herefter får man et nyt vindue hvor man skal skrive et projekt navn og et package navn. Hvis man også downloadede Google App Engine står den automatisk til, at man gerne vil bruge det i

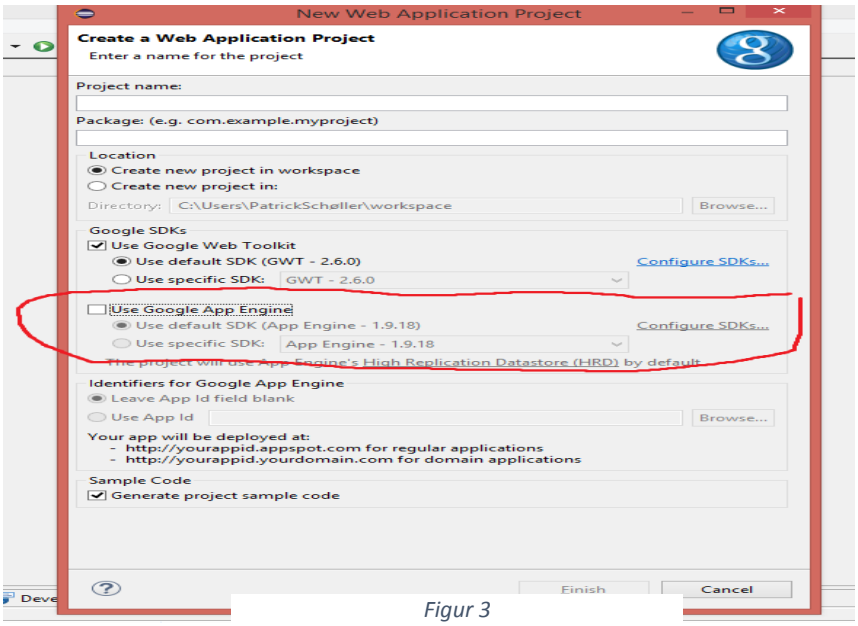


Figur 2 oprettelse af nyt GWT projekt

projektet, men hvis man gerne vil have muligheden for at bruge en database, som skal kaldes via de asynkrone kald, som i vores tilfælde, kan man med fordel slå Google App Engine fra.

Når man har gjort dette er der ikke mere man skal gøre andet end at trykke Finish, hvorefter der bliver oprettet et simpelt projekt som man kan arbejde videre ud fra.

Figur 3 viser det vindue der kommer op når man vil oprette et nyt GWT projekt. Den røde ring viser, hvor man skal slå Google App Engine fra.

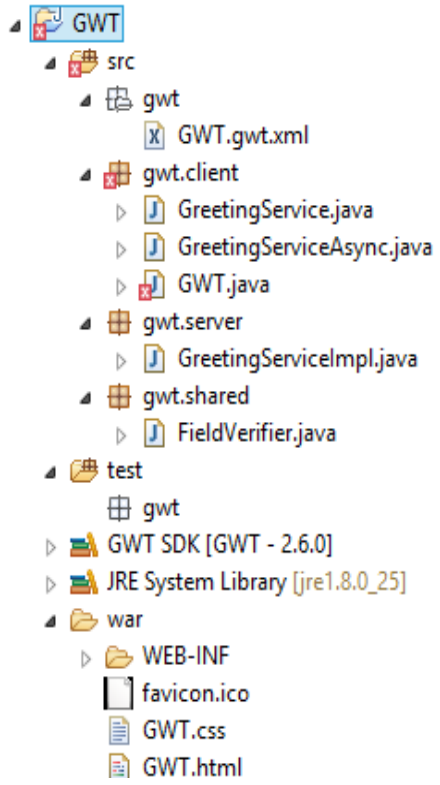


Figur 3

Når man har oprettet GWT projektet får man et projekt der indeholder 4 pakker i source filen.

De 4 pakker er:

- En GWT-pakke som indeholder XML-filen for projektet
- En klient pakke som indeholder de asynkrone kald og Java filen som er der, hvor man kører projektet fra via ”onModuleLoad” metoden.
- En server pakke, som indeholder Java koden til det der sker, når de asynkrone kald får fat i serveren, og det er her man kan kontakte en database.
- En shared mappe som indeholder en Java fil, som bruges på klienten til at tjekke om de navne, der bliver skrevet i feltet er lange nok.



Figur 4 hvordan standard GWT projekt ser ud

Ud over det er der også en war mappe, som indeholder css koden og html koden for projektet.

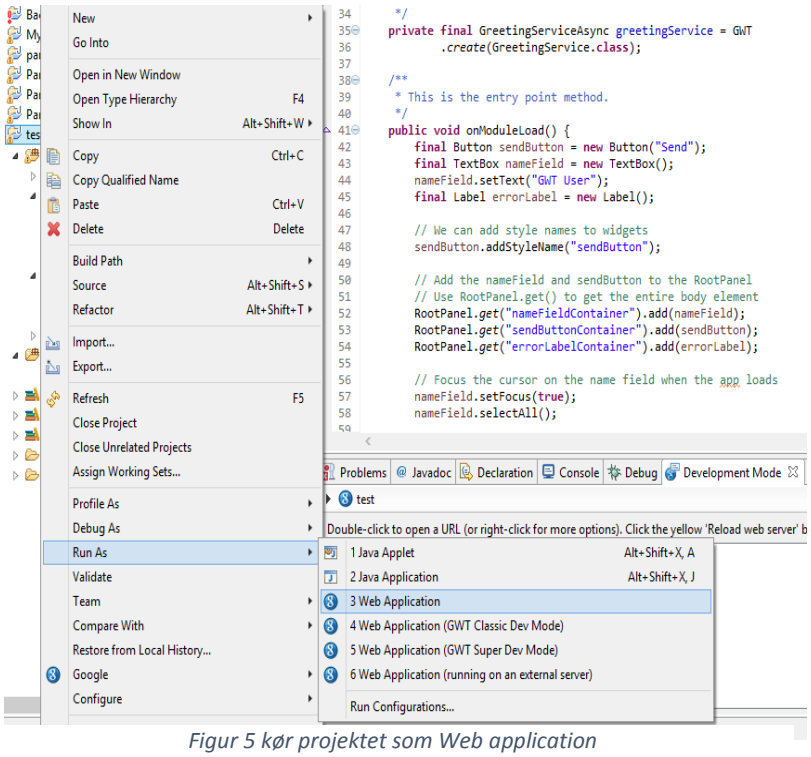
Css filen indeholder koden til udseendet for elementerne der bruges på siden.

Projektet ser ud som på figur 4.

Når man skal køre projektet, især for første gang, skal man højre-klikke på selve projektet og gå ned i ”Run as”. Her vælger man at køre det som ”Web Application” som vist på figur 5.

Efter første gang kan Eclipse, for det meste, huske hvad man har kørt projektet som, og derfor kan man trykke på Run knappen herefter.

Hvis man har nogle bestemte versioner af GWT, Eclipse og Java er man nødt til at kører projektet som ”Web Application (GWT Super Dev Mode) ” som er punkt nummer 5 i listen under ”Run as”.

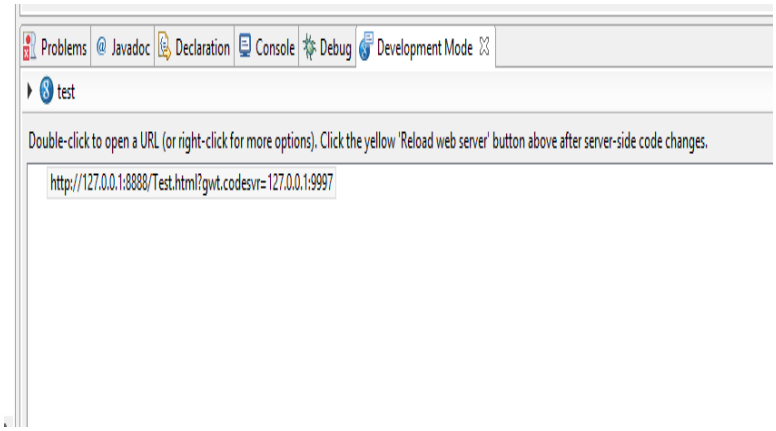


Figur 5 kør projektet som Web application

Når projektet kører, får man et link i konsollen som vist på figur 6.

For at køre projektet skal man dobbelt klikke på linket, og så vil det åbne i den internet browser som man har som standard.

En anden måde at køre det på, er ved at højre-klikke på linket og trykke ”open with” og så vælge den browser man gerne vil kører det i.



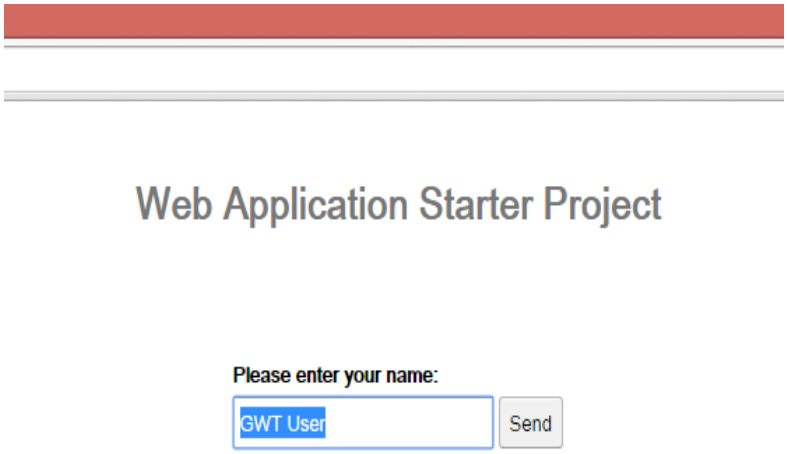
Figur 6 link til kørsel af porgrammet

Vi kørte projektet i Google Chrome, og første gang projektet kører, bliver man bedt om at hente et plugin for GWT til browseren som vist på figur 7.

Figur 7 plugin til Chrome

Når man har hentet det plugin får man vist projektet.

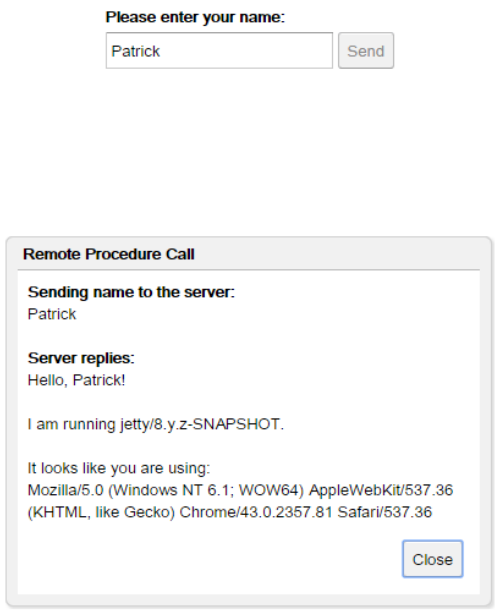
Når projektet kører ud fra den automatisk generede kode får man en hjemmeside op, som vist på figur 8, med overskriften ”Web Application Starter Project”.



Figur 8 Standard startsiden

På siden er der en tekstboks med overskriften ”please enter your name”.

Når man har skrevet et navn i tekstboksen kan man trykke enter eller trykke på knappen til højre for boksen.



Figur 9 popup vindue

Når man gør det bruger GWT RPC kald, det vil sige de asynkrone kald, til at lave et popup vindue.

I det popup vindue står der hvilket navn der er sendt til serveren og serverens svar.

Serverens svar er ”Hello” og navnet der er sendt og så skriver den hvilke versioner af de forskellige internet browsere, man bruger.

Dette er vist på figur 9.

Note på debugging i GWT

Denne note på GWT omhandler debugging.

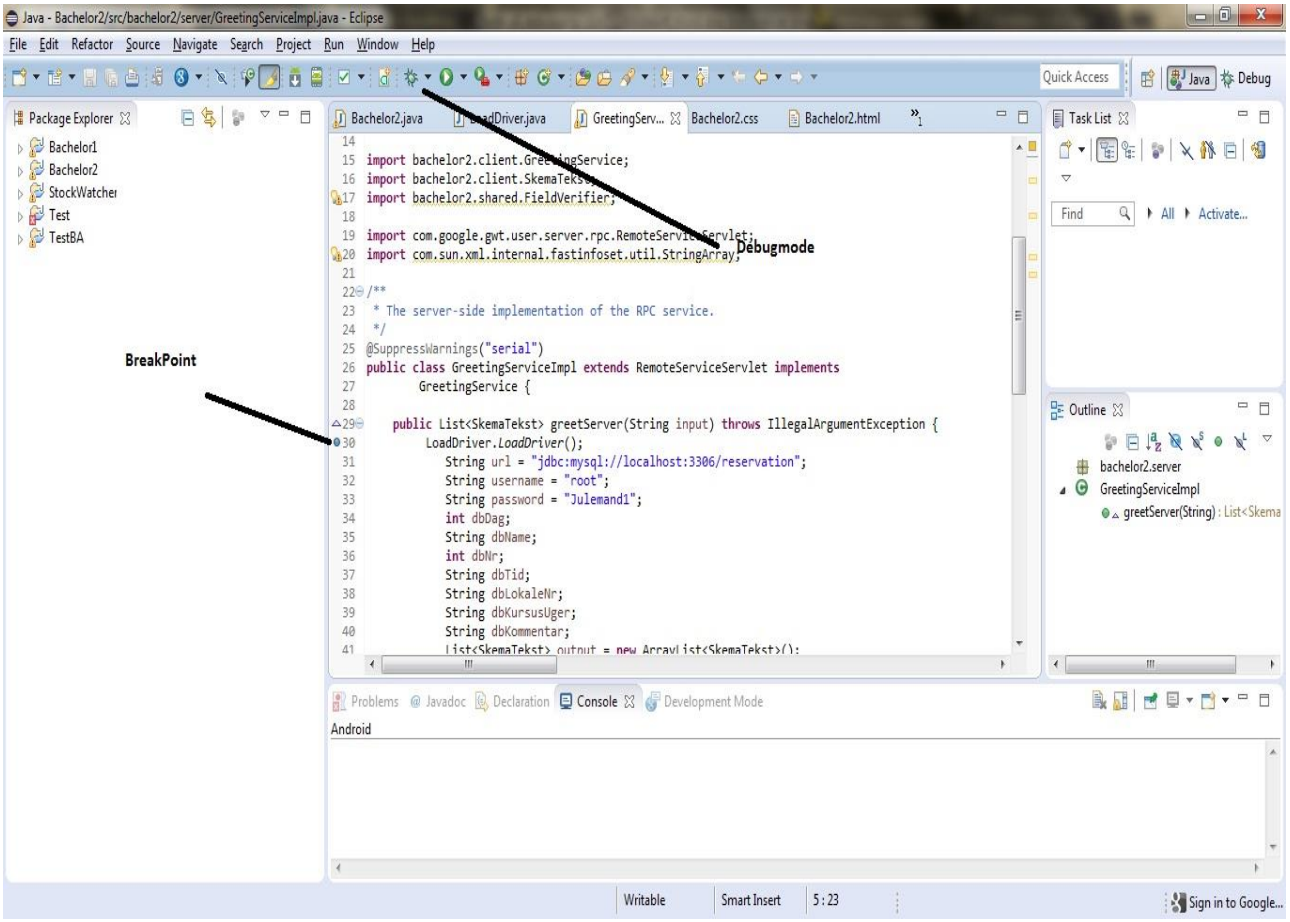
Systemet der er testet på er et Windows 7, 32-bit system.

Vi kører GWT version 2.6.0 og kører i Eclipse Luna og JDK 1.7.

Når man skal debugge i GWT er der to måder der kan bruges alt efter om man skal debugge server- eller klient-siden af ens program.

Hvis man skal debugge server-siden gøres dette ved hjælp af den indbyggede funktion som allerede ligger i Eclipse.

For at gøre dette sættes et breakpoint der, hvor man ønsker at debugge, og så afvikle programmet i debugmode.

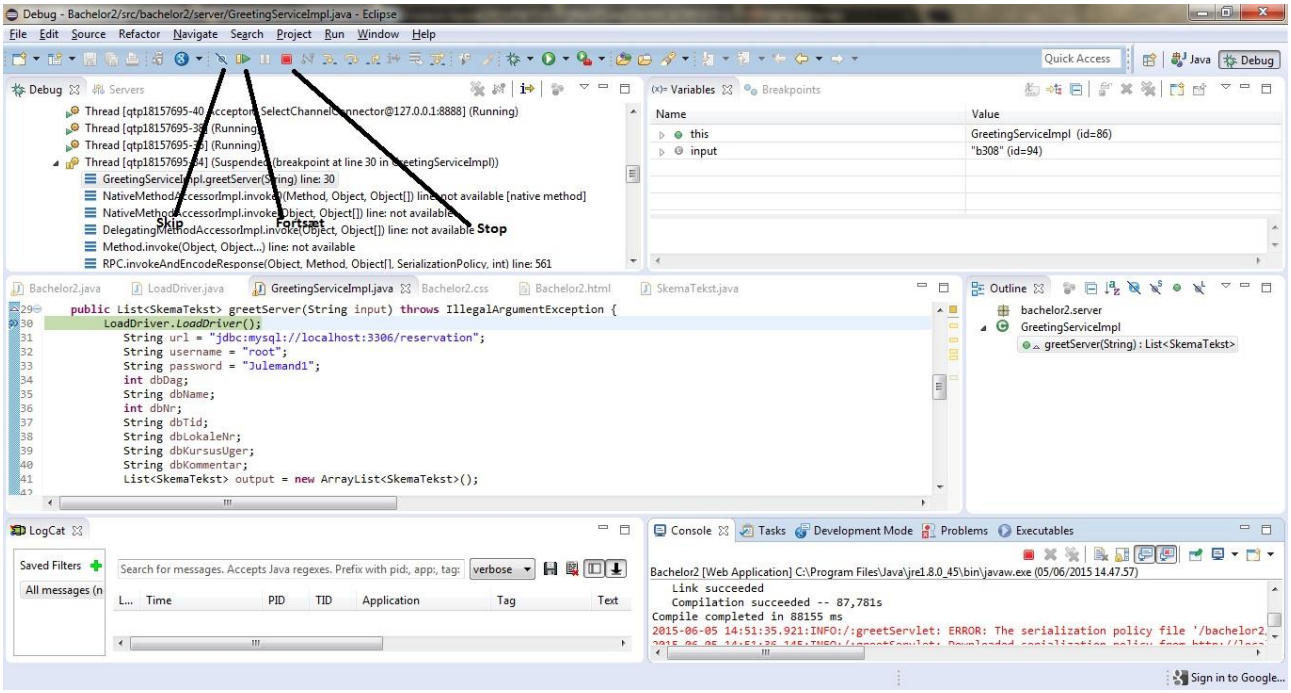


Figur 10 Debug og breakpoint

Enten ved at trykke på knappen i værktøjslinjen (se figur 10), eller ved at højreklikke på projektet og derefter vælge debug as Web Application (GWT Super Dev Mode).

Når man på siden foretager en handling der udløser et breakpoint skal man tilbage i Eclipse for at foresætte.

Et muligt billede på det kunne være

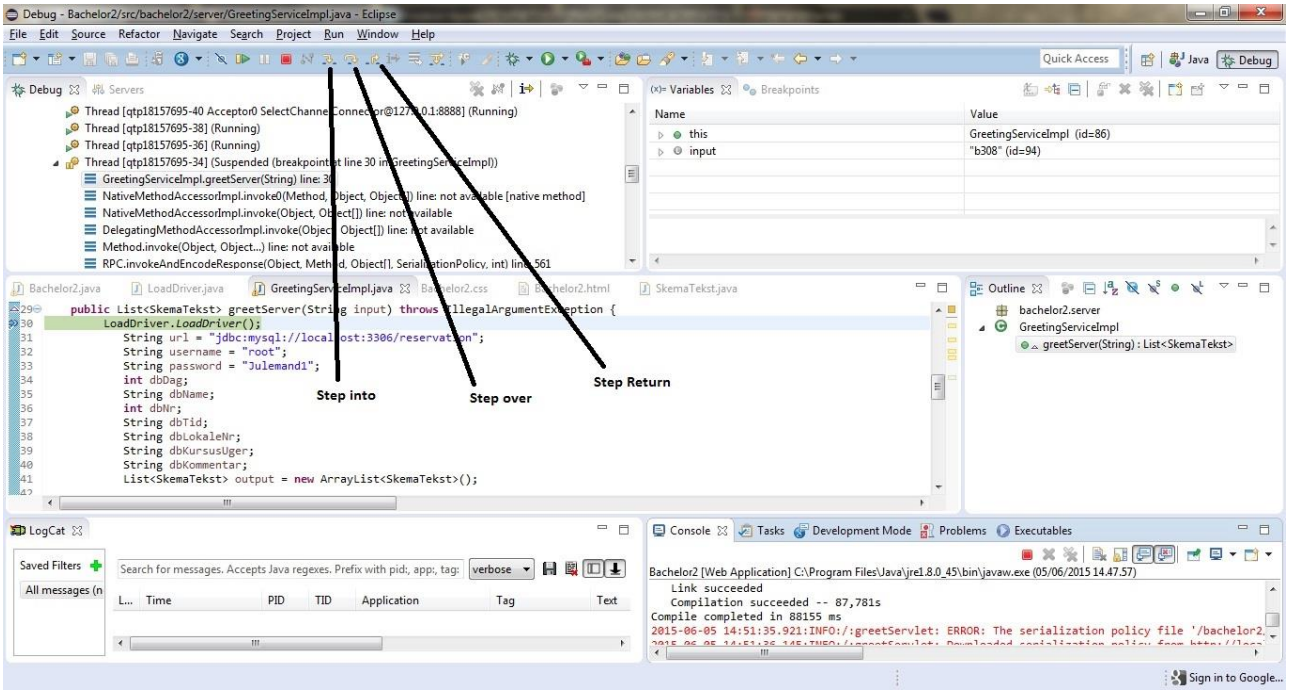


Figur 11 Ramt Breakpoint i Eclipse

Man har nu forskellige muligheder. Man kan droppe alle breakpoints, forsætte programmet (hvis man har mere end et breakpoint kører programmet kun videre til det rammer det næste), eller stoppe programmet.
Derudover er det forskellige funktioner til at dykke ned i koden.

Den ene mulighed er ”step into” (F5) hvor man går metoden igennem skridt for skridt, og også kalder de andre klasser der bliver brugt. Det er rigtig godt i forhold til at forstå detaljerne i programmet, men har desværre den ulempe, at den også går ned i de indbyggede biblioteker, det sker hvis det er et kald der bruger nogle af de indbyggede metoder.
En anden mulighed er at bruge ”step over”(F6) som går alle de skridt igennem som gør noget i den funktion der debugges, og udfører andre funktioner uden at dykke ned i dem.

”Step return” (F7) bliver brugt til at komme ud af en metode igen.

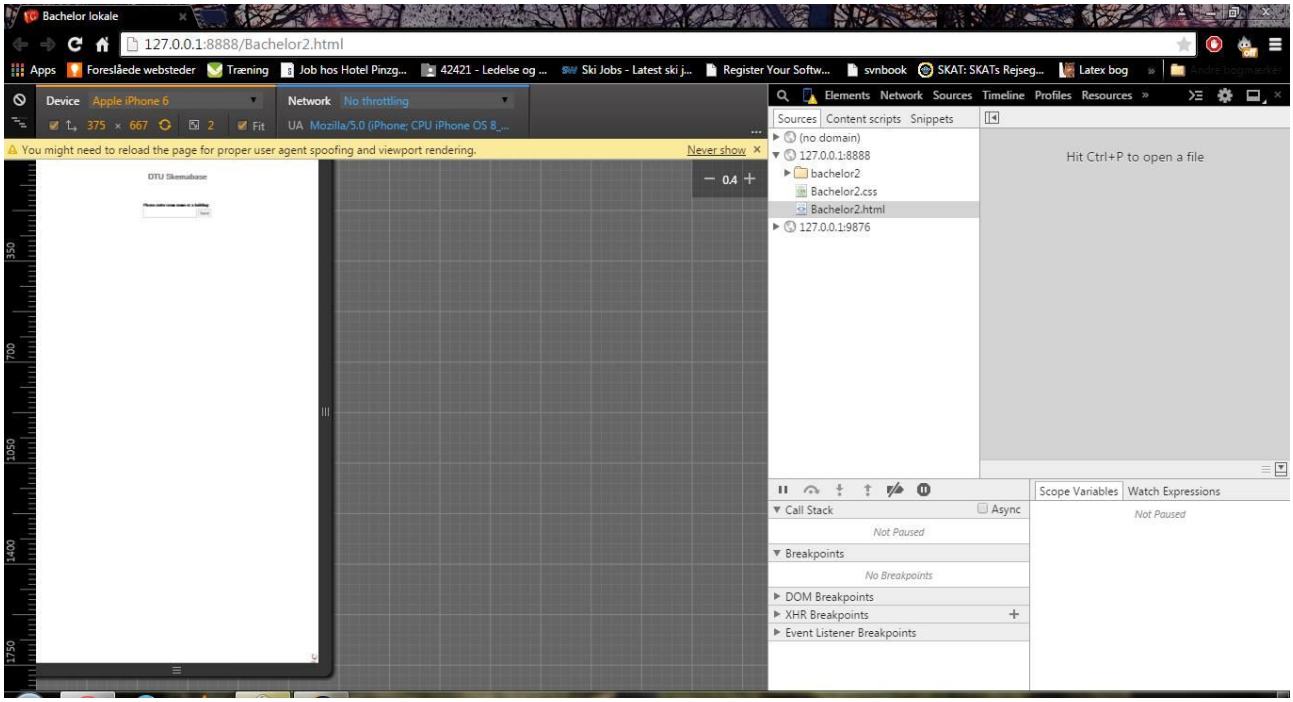


Figur 12 Markering af funktioner i debugging

For klient-siden er det lidt anderledes.

Her starter man med at køre projektet normalt. Altså ved at sige run as Web Application (GWT Super Dev Mode), og åbne linket i Chrome.

Herfra trykker man F12 og resultatet skulle gerne ende med at ligne figur 13

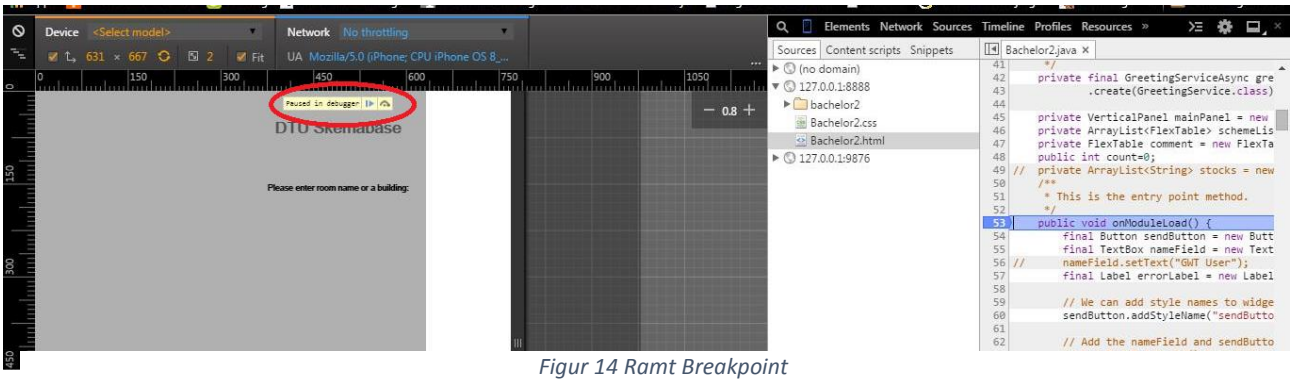


Figur 13 Når man åbner debug i Chrome

Hvis ikke det ser sådan ud når man har trykket F12 skal man gå ind under sources, hvilket skulle afhjælpe problemet.

For at finde de Java filer man gerne vil debugge skal man åbne den nederste mappe, i dette tilfælde 127.0.0.1:9876 og finde den mappe der hedder sourcemaps, som er der hvor klient mappen ligger.

I klient mappen kan man finde den fil man ønsker at debugge og så sætte et breakpoint som man også gør i Eclipse.



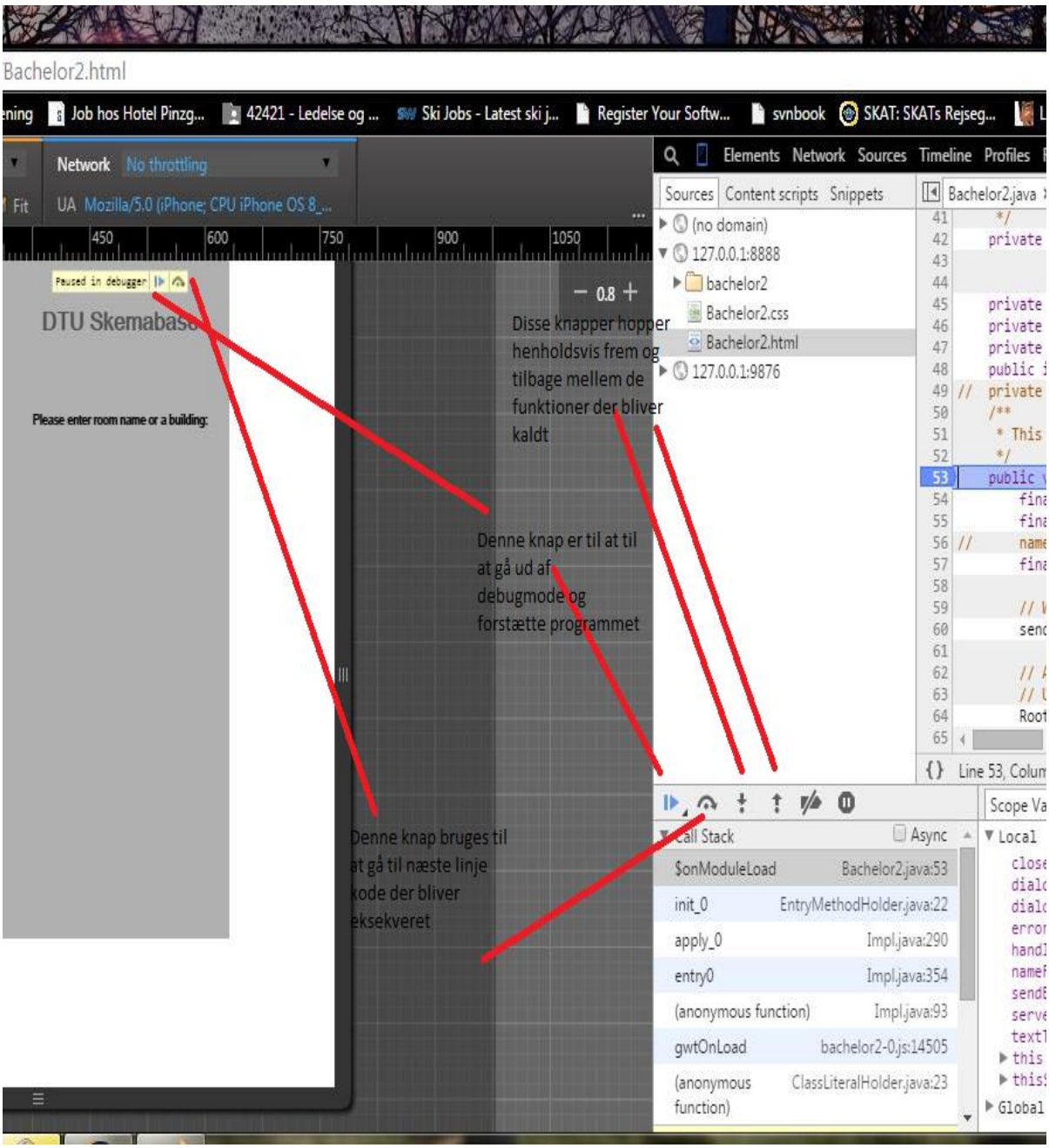
Figur 14 Ramt Breakpoint

Vi har her valgt at sætte et breakpoint på funktionen onMuduleLoad() hvilket betyder vi bliver nødt til at reloade hjemmesiden for at se om det virker, se figur 14.

Der skulle nu meget gerne stå ”paused in debugger” som i den røde cirkel.

Figur 15 viser debugmuligheder i Chrome

De to sidste knapper er henholdsvis ”fjern breakpoint” og ”stop ved exception”

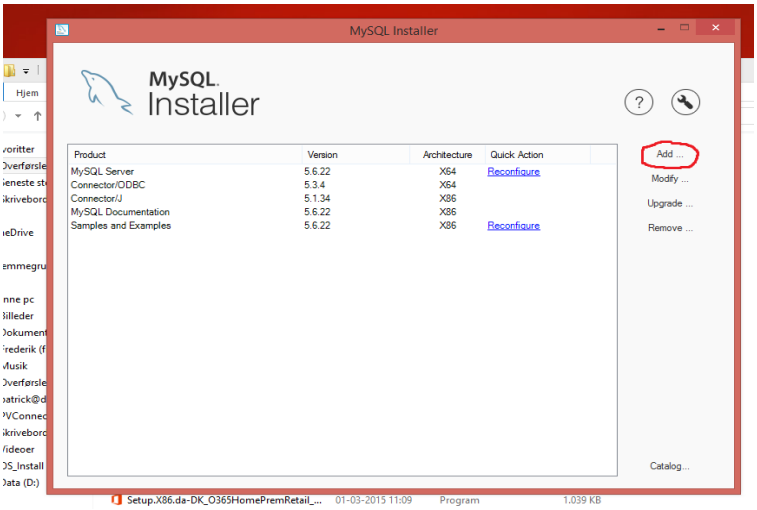


Figur 15Funktioner i debugging

Opsætning af MySQL database

For at kunne oprette en MySQL-server skal man gå ind på deres hjemmeside som er mysql.com.

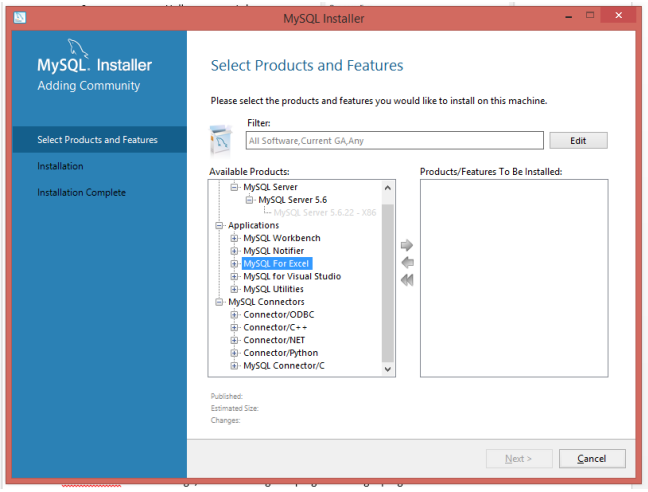
På siden skal man gå ned i bunden og finde downloads, og vælge MySQL-server. Dette gør at man kommer ind på en side, hvor man kan downloade en MySQL installer som bruges til at hente de nødvendige filer og installere dem.



Figur 16 MySQL installer

Når man har installeret installeren og kører programmet skal man vælge knappen Add, som vist på figur 16.

Derefter kommer man til en side hvor man skal vælge de MySQL produkter man skal bruge og



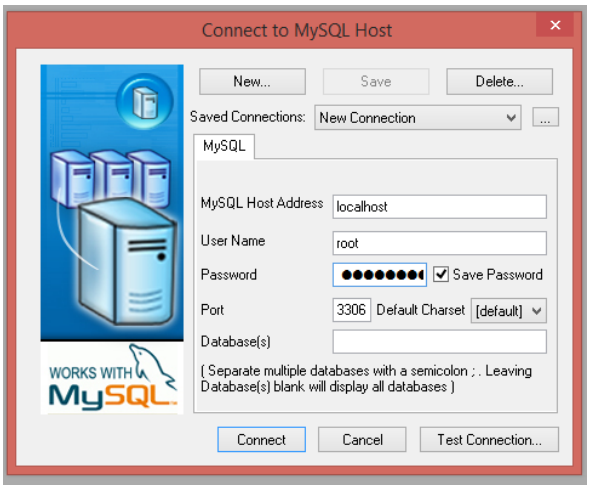
Figur 17 Valg af produkter MySQL

hvilke connectorer man skal bruge, hvilket afhænger af det programmeringssprog man bruger.

Vi skulle bruge den fil der lå under server, excel applikationen og filen connector/J for at vi kan bruge det i forbindelse med Java.

Når man har valgt de filer der skal bruges går man videre, og beder programmet om at downloade filerne og installere dem.

Når de er installeret kan man gå ind og sætte serveren op. Her vælger man hvor tilgængelig den skal være, og hvad port den skal bruge. Derefter vælger man et brugernavn og password for administrator og når man har valgt de ting kan man få serven til at starte.



Figur 18 Forbind til lokal server

Da vi havde fået serveren op at køre havde vi brug for at kunne lave en database og vedligeholde denne. Det gjorde vi ved hjælp af værktøjet SQLyog som bruges til at forbinde til serveren, og derfra gør den det simplere at lave nye databaser og tabeller på serveren.

Når man kører SQLyog for første gang skal man oprette en ny forbindelse for at kunne forbinde til serveren.
Dette gøres ved at bruge den rigtige MySQL adresse hvilket er ”localhost” hvis det bare er en lokal server på computeren og ellers er det linket til serveren man skal bruge.
Samtidig kan man bruge brugernavnet ”root” hvis MySQL-serveren er lokal. Hvis ikke skal man bruge et brugernavn på en af de oprettede brugere til serveren og en tilhørende kode. Hvis man bruger ”root” er det koden til administratoren man skal bruge.

Eksempel er vist på figur 18.

Når man har forbundet til serveren kan man se alle sine databaser på serveren til venstre og i databaserne kan man oprette eller ændre tabellerne.

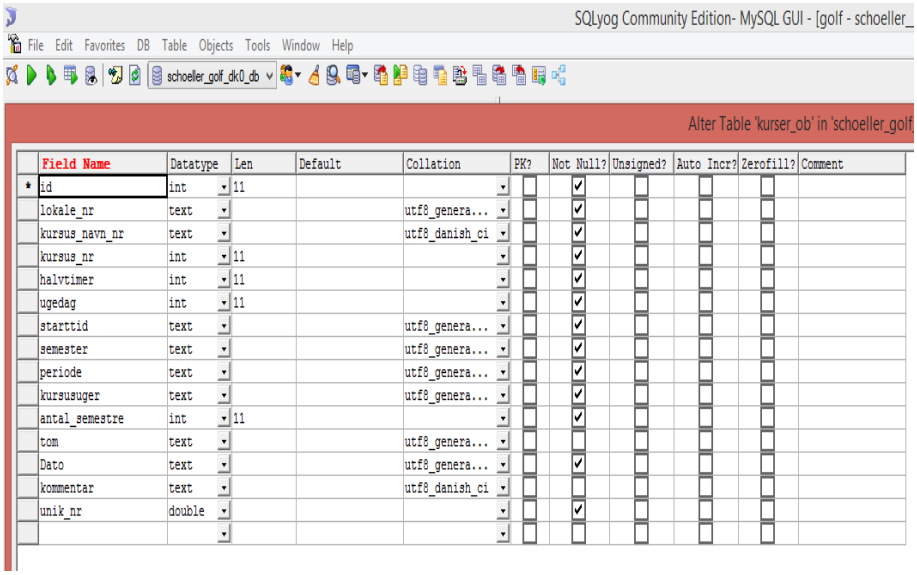
Man har også muligheden for at lave nye databaser, hvis det er en lokal server ellers kan man være begrænset af hvad den server man har adgang til tillader.

I vores database har vi oprettet to tabeller en til hver af de filer vi arbejder med.

Tabellerne er oprettet ved at vi går ind og siger hvilke elementer de skal indeholde og hvad typer de elementer skal være som vist på figur 19.

Da tabellerne var oprettet kunne vi gå ind og højre klikke på den tabel vi skulle bruge og sige importer fra csv-fil og hente dataene direkte fra de Excel- /csv-filer vi havde fået udleveret til projektet.

Et af de problemer vi stødte på i forhold til typer og sprog var, at de filer vi hentede var gemt med sprogtypen ANCSII og lige meget hvad sprogtype vi valgte i tabellen stoppede den enten med at hente resten af navnet når den nåede bogstaverne æ, ø eller å, eller også skrev den det ind som spørgsmålstegn. Vi fandt frem til, at hvis vi åbnede filerne ved hjælp af notepad++, kunne vi ændre sprogtypen til utf8, og det løste vores problem. Derefter kunne vi få teksten hentet rigtigt ind og bruge det i vores program.



Figur 19 Opsætning af tabeller

Forbindelse til Databasen med Java koden

Analyse

Det første problem vi skal løse er at finde en måde hvor vi kan lave et program, der kan forbinde til en MySQL-database, for at vi kan få fat i de informationer vi skal bruge senere i projektet.

Første punkt er at kunne få kontakt til serveren og andet punkt er at få fat i specifikke informationer, der ligger i databasen.

Det vi skal være opmærksomme på er, at vi i første omgang skal lave et program hvis eneste opgave er at forbinde til databasen, og derefter skal vi få det til at fungere i samarbejde med de asynkrone kald der bliver brugt når man arbejder med GWT.

Grunden til, at vi starter med et program der skal forbinde til databasen, uden at bruge de asynkrone kald, er at vi vil sikre at kaldene til databasen fungerer. Når dette er på plads kan man få det til at virke med de asynkrone kald. Dette gøres for at man nemmere kan se om det er forbindelse til serveren eller de asynkrone kald der fejler.
Det vil sige, at når vi har lavet et program som kan forbinde til databasen, skal vi have det ind som serverkoden i vores GWT projekt og så tilpasse de asynkrone kald så vi får de rigtige informationer tilbage til klientkoden.

Vi skal også være opmærksomme på at lave det på en simpel og overskuelig måde, så det er nemt at definere hvilket informationer vi skal have fat i og hvilke tabeller der skal bruges.

Diskussion

Der er flere måder man kan løse dette problem på og hver løsning har to trin.

Det første trin i begge løsninger er at man skal hente en ” JDBC driver” som sørger for at man kan forbinde til en MySQL server.

Der er to måder vi kan lave koden, der kan hente driveren.

Den første løsning er at vi skriver koden i toppen af klassen der skal forbinde til databasen og dermed holder al koden samlet i en klasse.

Den anden løsning er at vi laver en klasse kun til den del af koden der skal hente JDBC driveren, og dermed bare skal kalde metoden inde i den klasse som skal forbinde til Databasen.

Fordelen ved den første løsning er, at alt koden er samlet et sted hvilket gør at man ikke skal kigge flere steder efter det der skal rettes, og det kan være en fordel hvis man ved, hvor man skal kigge i koden for at løse de problemer man kan støde på. Eller når man skal skifte driver.

Fordelen ved den anden løsning er, at man nemmere får overblik over hvilken del af koden, der er i bestemte klasser, da hver klasse kun vil indeholde den kode der skal til for, at de løser deres opgave.
Det vil gøre det nemmere at overskue koden, og det vil dermed også gøre det nemmere at gennemse koden, og dermed hurtigere at finde de fejl der kan være.
Det kan også medføre at det er nemmere at rette eventuelle fejl da det kode man arbejder i ikke indeholder andet end det der lige løser den opgave man arbejder med.

Det andet trin er at lave kode som kan bruge JDBC driveren til at forbinde til Databasen.
Dette trin vil stort set være det samme alt efter, hvad man vælger som første trin. Den eneste forskel vil være om man henter driveren ved at kalde en metode fra en anden klasse eller om man vælger, at den del af koden skal ligge i toppen af funktionen.

Man skal derefter skrive alle forbindelses-informationerne, som brugernavn, kodeord, link til serveren og database navnet.

Når man har skrevet de ting ind skal man sikre sig at ens projekt har en MySQL-connector-java fil, da denne skal bruges for, at man kan bruge Java kode til at forbinde til en MySQL-server.
MySQL-connector-java filen skal tilføjes til de biblioteker, som projektet har adgang til.

Når det er klaret skal man oprette en ”connection” altså en forbindelse til serveren ved hjælp at JDBC driveren.
Forbindelsen skal indeholde alle forbindelses-informationerne.

Sidste trin for at løse problemet vil være, at vi skal få programmet til at samarbejde med de asynkrone kald fra vores GWT projekt.
Det kan gøres på to måder, den ene måde er, at vi bruger de filer der er oprettet i forvejen og derefter bare tilpasser kaldene. Dette sikrer at alle filerne er sat op på den rigtige måde.

Den anden måde er, at vi laver helt nye filer og laver vores egne asynkrone kald.

Design

Vi valgte at implementere løsningen, hvor vi har koden til at hente JDBC driveren i en klasse for sig selv.
Det gjorde vi fordi vi synes, at det giver den bedste mulighed for at overskue koden, og dermed også gør det nemmere at fejlfinde koden hvis det var nødvendigt.

Vi startede med at lave et program, som bare skulle forbinde til databasen ved hjælp af Java koden og JDBC driveren.
Det blev gjort ved at lave to filer. Den ene blev brugt til at hente JDBC driveren, og den anden blev brugt til at lave en forbindelse til serveren.

Andet trin i det projekt var at hente en MySQL connector til Java fra MySQL hjemmesiden, og derefter sørge for at programmet refererede til den jar fil som vi havde hentet.

Sidste trin i dette problem var at få det program vi har lavet til at samarbejde med de asynkrone kald i GWT projektet.
Det valgte vi at gøre ved at tilpasse de allerede eksisterende kald, hvilket vi gjorde fordi det var første gang vi skulle arbejde med asynkrone kald, og derfor var det en nemmere måde at gå til problemet. Desuden syntes vi, at det var smartere at bruge allerede eksisterende informationer i stedet for at skulle prøve at genskabe noget, som vi allerede havde.

Implementering

Vi startede med at lave et helt almindeligt Java projekt, da vores ide var at starte med at få forbindelsen til at virke og derefter arbejde på de asynkrone kald.

I projektet lavede vi to Java klasser. Den ene hed LoadDriver og den anden hed Connector.

Filen LoadDriver går ind og sætter linket til at hente JDBC driveren, og sørger for at det sker fejlfrit.

Filen Connector starter med at kalde LoadDriver for at hente JDBC driveren. Derefter går den ind og definerer linket til serveren, brugernavnet og kodeordet, der skal bruges for at forbinde til databasen.

Næste del er, at den går ind og definere den sætning, der skal bruges for at vælge elementer fra databasen og fra specifikke tabeller

Når det er klaret går den ind og opretter en forbindelse, og bruger sætningen til at få de elementer ud som vi har bedt om, og når alle elementerne er fundet går vi resultaterne igennem og opretter de objekter vi skal bruge til resten af vores program.

Da vi havde fået forbindelsen til at fungere valgte vi at bruge de allerede eksisterende asynkrone kald. Derfor tog vi vores kode fra Connector filen og satte ind i den kode man har fået i filen GreetingServiceImpl.java som ligger i server koden i GWT projektet.

For at få det til at fungere var vi nødt til at ændre retur typerne for metoden, og de steder hvor metoden blev kaldt. Herefter kunne vi få de asynkrone kald til at bruge vores Connector kode.

Det eneste der så manglede var at vi skulle lægge LoadDriver filen i server mappen i GWT projektet, og skrive kode i klient-koden der kunne bruge de informationer der nu kom tilbage fra de asynkrone kald.

Få vist et skema

Analyse

Den mest essentielle del af programmet er at oprette og vise skemaerne.
Disse er både væsentlige for dem, der skal lægge skemaerne i første omgang, hvorfor det også er vigtig at holde det simpelt og let overskueligt, da det kan give et hurtigt overblik over hvor der er dobbeltbookninger, og for brugerne som kan finde deres lokaler og placering af dem på campus.
Derfor er det vitalt at disse bliver lavet ordentligt og på en fornuftig måde. De filer der ligger til grund for vores skemaer består af kurser oprettet med bla. navn, dag, starttidspunkt, længde, kursusuger og ønsket lokale, og det er ud fra disse informationer at skemaet skal laves.

Det vi skal være opmærksomme på er, at vi i starten bare skal få vist et skema og derefter vise, at vi kan skrive ting ind i skemaet.

Når det virker, skal vi så fokusere på at bruge de informationer vi har fået fra databasen og bruge dem til at vise ting i skemaet, både der hvor der kun er enkelte bookninger, men også der hvor der er dobbeltbookninger, da det er vigtigt at vi kan vise begge dele.

Når det er løst vil næste fokuspunkt være at vi skal kunne vise flere skemaer og kunne søge flere gange.

Når vi skal vise flere skemaer skal vi være opmærksomme på, at vi skal vide hvilke skemaer der viser hvad. Det vil sige at vi skal sørge for at der bliver skrevet et lokalenummer på skemaerne og vi skal være opmærksomme på, at der kun skal vises flere skemaer, hvis man søger på en bygning, og ikke hvis man søger på et bestemt lokale.

I forhold til det med at søge flere gange skal vi sikre os, at vi opdaterer siden rigtigt. Det vil sige, at vi skal sørge for at programmet laver siden fra bunden, i forhold til skemaerne, da vi ikke skal vise de samme informationer med mindre de søger på samme lokale igen. Men det behøver man ikke tage højde for, hvis programmet altid virker ved at søge igen.

Derefter er der punktet med at kunne vise forskellige uger, og her skal vi være opmærksomme på, at det ikke er semesterugerne vi kan vise, men alle de uger der er i mellem semesteret starter og semesteret slutter.

Det er både fordi, at den fil vi har fået fokuserer på alle ugerne imellem start og slut og kun fokuserer på om kurserne forgår i den uge, men også fordi, at det kan give et bedre overblik over hele semesterets forløb, det vil sige at man også får overblik over ferier og helligdage som påvirker hvornår kurserne kører.

Derudover er det også en fordel for dem der gerne vil ind og se hvornår lokalerne er ledige, så de selv kan bruge dem.

Diskussion

Der er flere måder at oprette skemaerne på. En af måderne er at oprette det som en tabel direkte i html koden, dette kræver at man har alt den info som man ønsker at have på et skema, og så derefter opretter skemaet.
En anden måde er at lave skemaerne i Java delen af koden. Dette betyder, at man kan lave skemaerne løbende og ikke behøver at fastsætte en størrelse fra starten, men at man kan bruge en flextable, hvor rubrikkerne ændrer størrelse alt efter, hvor meget der står i dem.
Hvis man kun skal bruge et skema og kender størrelsen fra start kan man bruge html vejen, men da vi ikke ved hvor mange dobbeltbookninger, der ligger på et lokale kan det være en fordel at bruge en flextable med en dynamisk størrelse.
En af fordelene ved at lave skemaerne igennem Java koden er også, at man kan gemme alle informationerne der bliver brugt til at oprette skemaerne, og man kan gemme skemaerne med alle informationer. Dette vil gøre det nemmere for brugerne at kunne søge igen, da man så bare skal sørge for at skemaerne på siderne bliver fjernet, og så lave nye der passer til de nye søgekriterier.

Hvor man ikke vil kunne gøre det lige så nemt hvis man har lavet skemaerne i html koden da det ikke bliver lige så nemt at tilgå alle informationerne.

Samtidig bliver det også nemmere at lave flere skemaer når vi opretter dem ved hjælp af Java koden, da vi kan lave metoder i Java koden, som opretter et skema hver gang vi finder informationer der skal bruge et nyt skema. Hvor vi i html koden mere har brug for at kende informationerne, før vi skriver koden til at lave skemaerne, da de alle skal defineres med det samme.

Der er også flere måder at lave løsningerne til at kunne vise alle ugerne i semesteret.

Den ene løsning er, at man gemmer alle informationerne for hver enkelt uge i hver deres liste, som man derefter søger igennem når man når til den aktuelle uge.

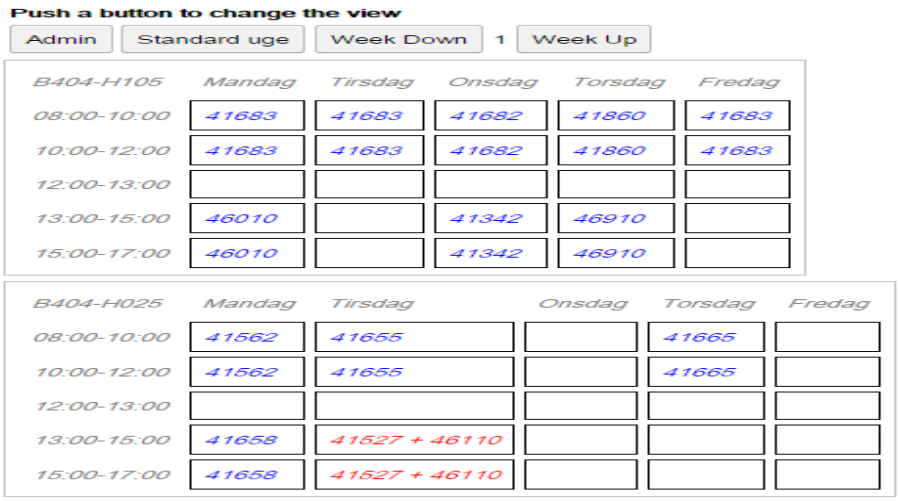
Fordelen ved denne løsning er at vi altid sørger for at de rigtige informationer for hver enkelt uge er samlet i en liste og derfor kan informationerne aldrig påvirke de andre uger. Men der skal skrives rigtig meget kode for at få den løsning til at fungere, da man skal ind og lave mange forskellige tjeks før man kan sikre sig, at alt fungerer på den rigtige måde.

Den anden løsning er, at når man tjekker om kurset skal skrives ind i et skema går man ind og tjekker om kurset er aktivt den uge og hvis det er, bliver det skrevet ind i skemaet, og hvis ikke springer man det kursus over for den uge.
Det kan gøres fordi vi sørger for at semesterforløbet er gemt i hvert objekt, som bliver oprettet når vi henter informationerne fra databasen.

Fordelen ved den løsning er at vi hver gang sikrer os, at kurset hører til på det skema i den uge og det kan gøres med mindre kode end den første løsning da man ikke skal lave tjek for hver enkelt uge og hver enkelt skema.

Design og implementering

Den måde vi gør det på er at vi opretter en liste af objekter, der indeholder de informationer vi har hentet fra serveren for den pågældende bygning eller lokale.
For alle elementer i listen tjekker vi om der allerede er oprettet et skema for det lokale som



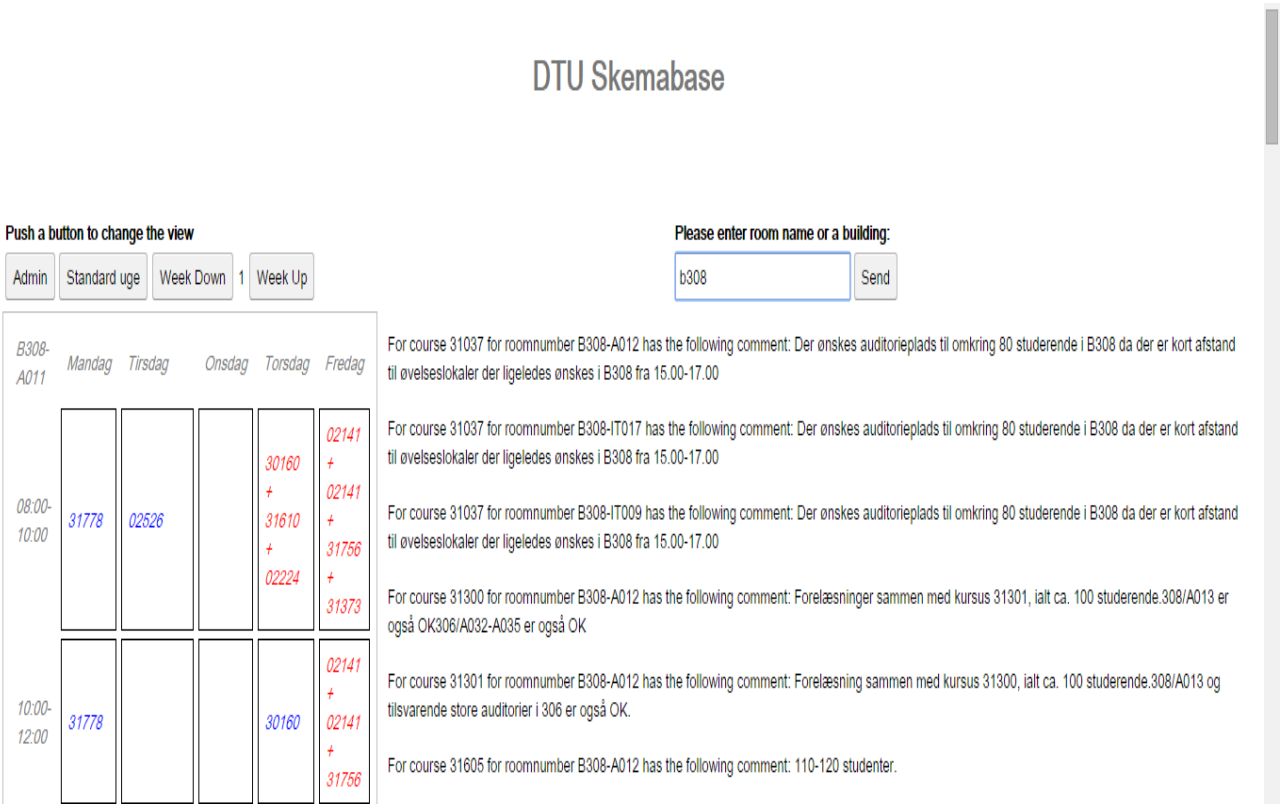
Figur 20 Skemaer

elementet vil bruge. Hvis dette er tilfældet henter vi det skema og opdaterer det med info for elementet, og ellers opretter vi et nyt skema og sætter elementet ind her.
Alle de lokaler vi har oprettet bliver gemt i en liste som er den vi tjekker imod, og bliver puttet på et panel, som er det vi bruger til at vise skemaerne på siden.
Hver gang vi opretter et nyt skema bliver det gjort ved hjælp af en funktion som opretter et standard skema med ugedag og tidspunkt, og så en masse tomme rubrikker, som er der hvor

kurserne bliver sat ind.

For hver gang vi får et nyt kursus tjekker vi kursets ugedag, starttidspunkt og længde for at finde den rigtige rubrik at sætte den ind i. Hvis rubrikken er tom sættes kurset bare ind og vi fortsætter til det næste. Hvis rubrikken ikke er tom beholdes det element, der allerede er samtidig med at det nye tilføjes. Når dette sker ændres skrift farven så det er nemt at registrere, hvis der er en dobbeltbookning når man skimmer hen over skemaerne.

Hvis et kursus har en kommentar tilføjes denne til et kommentarpanel, som ligger ved siden af skemapanelet.



Figur 21 Skemaer med kommentarer

Figur 20 og 21 er begge taget fra administrator delen af vores program, som er det der ligger til grund for skemalægningen. Som man kan se på figur 21 er der tre forskellige kurser der ønsker at bruge det samme lokale torsdag morgen. På figuren ses også nummeret på ugen man befinder sig i. Dette er fordi alle kurser også har en plan for hvilke uger bestillingen skal gælde. Nogle kurser

har måske kun brug for et auditorium i den første halvdel af semestret eller der skal tages højde for alle helligdagene i forårssemestret.

Måden vi sørger for at vi kan søge flere gange er, at vi har lavet en metode som vi kalder ”refresh”, og det den gør er at nulstille alle panelerne der er på siden, for at gøre dem klar til næste søgning.

Når den har nulstillet alle panelerne begynder den på søgningen, og tilføjer de informationer den finder frem til på de rigtige paneler.

Ugerne er ikke semester uger, men uger efter semesterstart. Grunden til dette er, at der er helligdage som f.eks. 2. påskedag, som gør at skemagrupper kan være i forskellige semesteruger.

Måden vi løser opgaven med at kunne vise alle semesterugerne er, at vi opretter 2 knapper der gør at vi kan ændre hvilken uge der bliver vist.

Når vi skal skrive kurserne ind i skemaerne tjekker vi først hvilken uge det er vi skal vise, og når det er tjekket går vi ind i og tjekker hvert objekts uge forløb, og ser om de er aktive i den uge som vi skal vise.

Hvis de er aktive skriver vi kurset ind i skemaet, og hvis de ikke er aktive springer vi dem over indtil de rammer en uge hvor de er aktive.

Muligheder for videre udvikling

Bookning af lokaler på siden

Analyse

Ideen med at kunne booke lokaler på siden, går ud på at man på et hvert tidspunkt i løbet af semestret kan gå ind og se hvilke lokaler der er ledige, og så bestille dem for et bestemt stykke tid, hvis man har brug for det.

Men også for at man kan gå ind og se hvilke lokaler der ikke er undervisning i, og derfor kan se om man kan sætte sig der og arbejde eller om lokalet allerede er optaget.

Samtidig vil man også kunne lave det om til at forelæserne skal gå den vej ind og booke lokaler til forelæsninger og grupperegning direkte på siden, da man så har et system der holder styr på det hele i stedet for at man skal hente en Excel fil ud fra campusnet og importere den ind i databasen, som booking systemet så skal ind og tjekke hver gang der bliver søgt på en bygning eller et lokale.

Man skal være meget opmærksom på at gøre det tydeligt visuelt, at man kan booke lokaler på selve siden og man skal sikre sig at når en tid bliver booket bliver den tilføjet til Databasen et sted, hvor den bliver taget med i betragtning når man tjekker for de forskellige bygninger eller lokaler, så det bliver vist i de rigtige skemaer så tiden ikke er ledig efterfølgende.

Et andet vigtigt fokuspunkt er at det skal være en let og overskuelig måde for brugerne at vide, hvordan de skal booke og hvilke informationer de skal give, da man skal sikre sig at det er de samme informationer der kommer fra Excel filen fra campusnet.

Diskussion

Der er flere forskellige måder man kan løse denne opgave på og der er to trin til hver løsning.

Det første trin er selve brugerfladen og hvordan brugerne skal finde ud af at booke en dag og give alle de rigtige informationer, og der er flere forskelige måder man kan lave det på, men vi har to specifikke som vi synes er dem vi vil diskutere.

Den første er at man kan lave en knap på siden, som er den samme lige meget hvad man har søgt på, og når man trykker på den knap kan man så få et vindue op, hvor man skal indtaste alle de informationer der skal bruges for at lave en bookning.

Den anden løsning er at man kan lave alle de felter der ikke er optaget på hvert skema til steder man kan trykke og så få et vindue op som allerede har skrevet dag og tid og lokale så man kun skal udfylde de resterende informationer.

Den første løsning er tilstrækkelig i de tilfælde, hvor man ikke har brug for at gå ind på et bestemt lokale og se om der er ledige tider.

Den anden løsning vil være mest brugervenlig og intuitiv, da man så kommer ind på det lokale man vil booke og så kan se om det er muligt. Hvis det er muligt kan man vælge den tid der passer en og så bare trykke på den.

Det andet trin til hver løsning omhandler den måde man opbevarer alle bookningerne på og igen kan det gøres på flere forskellige måder, men vi fokuserer på to løsninger.

Den første løsning kan være, at man tilføjer de nye bookninger til den samme tabel i databasen, som den der indeholder informationerne fra Excel filerne, det kan gøres ved at man tilføjer en del mere til tabellen, som holder øje med om bookningerne kommer fra Excel filen eller fra hjemmesiden, da man skal sikre sig at informationerne fra Excel filen skal kunne slettes hvis informationerne skal opdateres.

Den anden løsning er at man laver en tabel mere som indeholder alle informationerne der kommer fra hjemmesiden, og derfor holder informationerne fra Excel filen adskilt fra dem der er booket fra siden. Det vil gøre at man ikke risikere at miste bookningerne fra hjemmesiden når Excel filen er opdateret og skal indlæses i databasen på ny.

Med den anden løsning skal man så sørge for, at man også tjekker den tabel igennem når man søger efter bookninger.

Den første løsning vil være bedst, hvis man har svært ved at gennemsøge flere tabeller i databasen.

Den anden løsning vil være bedst i stort set alle tilfælde, så længe man kan gennemsøge flere tabeller, og da vi allerede gennemsøger to vil tre ikke være den store ændring.

Grunden til at den anden løsning er bedst i stort set alle tilfælde er, at man ikke risikerer at miste de data man har fået fra hjemmesiden, og derfor altid har muligheden for at opdatere informationerne fra Excel filen uden at være nervøs for at man mister andre data.

Design

Hvis vi implementerede bookning i vores program ville vi vælge den løsning, hvor hver ledige skematid er en knap man kan trykke på for at booke en tid. Dette er for at gøre det så brugervenligt som muligt.

Samtidig ville vi vælge den opbygning af databasen, hvor vi laver en separat tabel kun til at indeholde de bookninger, som kommer direkte fra hjemmesiden.

Der er flere grunde til vi mener, at dette er optimalt, den første grund er at man fjerner risikoen for at miste bookningerne fra siden ved at holde dem adskilt fra bookningerne som kommer fra campusnet.

Det vil også gøre det nemmere at gå ind og finde ud af om det en bookning for et kursus eller om det er en bookning, der er lavet efter semesteret er startet.

Grunden til at det vil være mere brugervenligt med at man kan trykke på de ledige tider i skemaerne for at booke dem er at man kan hjælpe brugerne med at udfylde flere af informationerne f.eks. dag og starttidspunkt.

En anden grund er at det er nemmere for brugeren at finde ud af om de tider de har brug for er ledige, når man alligevel skal ind og finde skemaet for det lokale man gerne vil booke.

Implementering

Når vi skal implementere denne løsning vil vi starte med at bruge den løsning med at have en knap til bookninger som ikke er afhængige af skemaerne, og så vil vi lave den separate tabel inde i databasen.

Grunden til, at vi vil starte med den knap er at det vil være den simple måde at få et bookning system op at køre, hvor vi ikke skal ind og ændre alt for meget i den allerede eksisterende kode, samtidig med, at vi kan få sat de asynkrone kald op, som skal bruges for at kunne tilføje bookningerne til databasen.

Det skal gøres ved, at vi får en knap tilføjet på samme måde som vores søge funktion fungerer og når vi har den knap skal vi lave en metode som kommer op med et vindue, hvor brugeren kan indtaste alle de data der skal bruges for en bookning.

Derefter skal der laves et asynkront kald, som skal forbinde til databasen og indsætte bookningen i tabellen.

Når vi så har sikret os, at vi kan booke tider, og at de bliver skrevet på den rigtige måde, så kan vi fokusere på at ændre skemaerne så hver af de ledige tider kan blive til knapper som skal bruges til at booke tider.

Det skal gøres ved, at vi går ind og ændre i nogle af de allerede eksisterende metoder, som bliver brugt til at generere skemaerne og til at udfylde dem, og derefter sørge for at når der bliver trykket på en knap inde i skemaet bruger vi knappens placering til at vide hvad dag og starttid er på den bookning brugeren vil have.

Derefter er det næste trin at bruge de metoder, som man lavede i første del til at bruge det asynkrone kald til at sende informationerne til databasen.

Se navn og link til kursusbasen.

Analyse

Ideen med at kunne se navnet er at gøre det muligt at holde over et af de bookede kurser og derved få navnet på kurset vist når man holder over et af elementerne i skemaerne.

Ideen med at linke til kursusbasen er at hvert element i skemaerne skal være et link, der går direkte ind til DTU kursusbasen, hvor man kan læse mere om kurset.

Det skal bruges for at gøre det nemmere for dem der bruger kursusbasen til at gå ind og se hvad der står omkring kurserne og få bedre overblik over hvad der er i lokalet.

Det man skal være meget opmærksom på er at gøre det tydeligt visuelt, at man kan trykke på et kursus og så komme ind på kursusbasen og læse om kurset. Samtidig skal man også sikre sig at hver gang man holder musen over et kursus bliver navnet skrevet på en måde, så det ikke er i vejen men tydeligt at se.

Et andet vigtigt fokuspunkt er, at det skal være let for brugerne at komme ind på kursusbasen uden at det påvirker den hjemmeside de har åben for at se skemaet. Det vil sige, at det skal åbne et nyt vindue som viser kursusbasen.

Diskussion

Der er flere måder at løse dette problem på.

Den første løsning er, at vi laver vores skemaer om til, at de kan håndtere, at der bliver tilføjet objekter til dem i stedet for, at det bare er tekst der bliver sat ind i skemaerne.
Det vil gøre, at vi altid har adgang til informationerne som kursus-nummer, -navn og alle de andre informationer som vi har fået fra Databasen.

Når man så har fået den til at modtage objekter skal vi lave objekterne om til links, som skal fører ind på kursusbasen og finde hen til det rigtige kursus.

Derefter kan programmet kører på samme måde som det allerede gør, bare med de nye skemaer og muligheden for at se navnet på kurset og at klikke på linket, der leder til kursusbasen.

Den anden løsning vil være at lave hele strukturen om i programmet og lave nye objekter, der passer bedre til at blive vist på siden, og som indeholder et link som leder til kursusbasen for det kursus som objektet beskriver.
Derefter skal vi så lave skemaer der kan bruge de objekter, og som kan vise navnet når man holder musen over kurset, og når man trykker på det skal det bruge linket der er gemt.

Fordelen ved den første løsning er, at vi ikke skal lave hele stukturen om i vores program.
Ud over det er det også en fordel at kunne bruge vores objekter, som det der bliver skrevet ind i skemaerne, da vi så har mulighed for at få fat i alle informationerne for kurserne, og derved har

nemmere ved at give flere informationer, hvis der er andet folk gerne vil vide omkring kurset og ellers så vil linket gøre at de kan få de sidste informationer som står i kursusbasen.

Fordelen ved den anden løsning er, at vi har alle informationerne gemt på en måde, som vi ved kan passe ind i de skemaer vi opretter, samtidig vil vi kunne sikre os, at linket er det rigtige, og at det virker på en anden måde end vi kan i den første løsning.

Det vil også gøre det nemmere at kunne kalde linket ved hjælp af nogle metoder, hvis vi har det gemt i objekterne, så det ikke har noget med skemaerne at gøre med mindre folk trykker på et objekt.

Man kan tilpasse de allerede eksisterende objekter til at kunne indeholde flere informationer og dermed undgå at skulle ændre hele strukturen men vi ville stadig skulle ind og ændre måden vi henter informationer, og hvordan vi opretter objekterne.

Fordelen er, at vi allerede opretter objekter og derved kan tilpasse objekterne og måderne vi kan få fat i informationerne og hvordan vi kan kalde et link.

Design

Hvis vi skulle implementere det med at kunne holde musen over et kursus og se navnet og kunne trykke på kurset og så blive ført hen til kursus basen, så ville vi vælge den løsning, hvor vi udvider de allerede eksisterende objekter og laver skemaerne om til at kunne håndtere objekterne og vise dem som links.

Samtidig vil vi gå ind og sørger for at når vi opretter objekterne vil vi kunne tilføje ekstra informationer, som vil kunne lave linket til kursusbasen og derved sørge for, at det er det rigtige link og gøre det nemmere at bruge linket.

Der er flere grunde til, at vi synes det er det optimale.

En af grundene er at det vil være nemmere for os at arbejde med og samtidig vil det kræve meget mindre arbejde at implementere i forhold til, hvis vi skal ændre hele programmets struktur.

En anden grund er, at løsningen vil fungere lige så godt som den løsning, hvor vi ændrer hele programmet, da skemaerne i sidste ende vil fungere på samme måde.

Den eneste grund til, at det ville kunne betale sig at skulle lave hele programmets struktur om vil være i det tilfælde hvor der er problemer med at de elementer man bruger i koden ikke passer overens med det der kan bruges som objekter i skemaerne.

Implementeringen

Når vi skal implementere denne løsning ville vi starte med at gå ind og tilpasse de objekter vi allerede opretter til, at de også vil kunne indeholde informationer om links og andre informationer vi synes er nødvendige for denne løsning.

Derefter vil vi gå ind og se på typen af skemaerne.
Da de for tiden er oprettet som flextables vil vi være nødt til at finde en type, der kan modtage andre elementer.

Flextables som vi bruger lige nu kan kun modtage tekst og kan derfor kun virke som den gør fordi vi laver alle de informationer der skal skrives ind om til Strings.

For at vi vil kunne bruge objekterne i skemaerne skal vi først finde en type for skemaerne der kan modtage objekterne eller bare de vigtigste informationer.

Når det er klaret skal vi så gå ind og sørge for at hvert element kan holdes over så man kan se kursusnavnet, og det vil gøres ved at sikre, at elementerne har en ”Hoover” funktion.

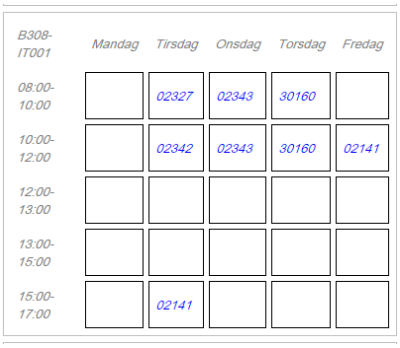
Sidste punkt som skal ændres er, at hvert element skal fungere som links, og det vil vi gøre ved at gøre det muligt at trykke på hvert element i skemaerne og når man trykker på dem vil vi kalde nogle metoder, som skal bruge linket fra det objekt man har valgt.

Gennemgang af hjemmesiden

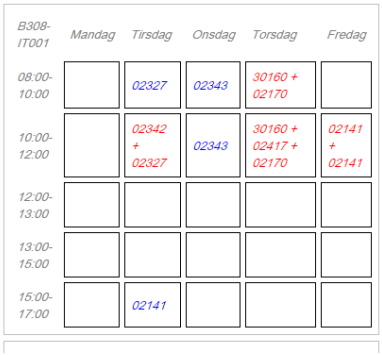
Når programmet startes kommer der en side op med nogle knapper og et tekstfelt.
I tekstfeltet skrives enten navnet på et rum eller en bygning som man ønsker at finde info på.
Når siden åbnes er den sat i standardmode hvilket betyder at det view man får op har skemaer uden dobbeltbookinger samt et kort der viser hvor på campus vi er.
Herfra har man mulighed for at for at lave en ny søgning, ændre til adminmode eller skifte uge.

For alle lokaler på campus Lyngby gælder det, at de starter med b efterfulgt af tre cifre som starter med 1-4, det giver f.eks ”b308” som er en bygning. Enten kan man søge på det eller hvis man ønsker et specifikt lokale kan man skriver mere og kun få skemaet for det ene lokale som ”B404-H105”.

Alle skemaerne har navnet på det lokale de tilhører, ugedagene og 5 tidsrubrikker. Kurserne er oprettet i skemaerne med kursus nummer. Figur 22 og 23 viser det samme skema i standard- og adminmode



Figur 22 Skema i Standard



Figur 23 Skema i admin

Skifter man til adminmode får man vist de samme skemaer, men nu er der også dobbeltbookninger og billedet er skiftet ud med kommentarerne for de viste skemaer. Herfra kan man så se, hvad der skal ændres for at skemaet går op.
De andre knapper går enten en uge frem eller tilbage, på den måde kan man se om der er uger hvor der ikke er undervisning, eller bare generelt om der er lokaler der ikke bliver brugt som man kan bruge, hvis man får brug for det i løbet af semestret.

Kodenær gennemgang

Hvis vi kigger mere kodenært på programmet er det første der sker når vi starter siden, at vores program kører funktionen ”onmoduleload”, svarende til en main metode i andre programmer. Her opretter vi vores knapper, vores textfields og textlabels, der er på siden fra starten. Det hele bliver oprettet ved hjælp af indbyggede biblioteker, som man også kender det fra bl.a. C#.
Man opretter objektet som en button (en knap), herfra kan man så give den en css style, og så fastgøre den til rootpanel, som er det laveste niveau vi kan sætte ting på, ved hjælp af en container. Det er disse containers id’er som vi kalder i html koden for at få dem vist. Det er samme fremgangsmåde, der også bliver brugt til textfields og textlabels.
Herfra kan man så skrive noget ind i tekstfeltet og trykke enter eller på send knappen. Gør man det bliver ”send to server” funktion kaldt. Her er først et tjek for at se om det er et lovligt kald. I vores program har vi valgt at have den til minimum 4 karakterer da det er hvad der skal bruges for at kunne bestemme en bygning. Så tjekkes der via de asynkrone kald om der er forbindelse til serveren, hvis ikke kommer der et popup vindue med en fejlmeddelelse, hvilket sker ved metoden ”onfailure”.

Ellers sendes den værdi der er søgt på til serveren, og såfremt der er noget for det pågældende lokale eller bygning, får vi en liste af skematekst som er det element vi bruger til at holde styr på informationerne vi får fra serveren. Hvis der ikke er noget på serveren, der matcher vores søge-værdi kalder vores ”onsucces” vores ”refresh” funktion, som fjerner det der er på siden og lader den stå tom. Får vi noget tilbage fra server kaldet bruger vi dette til at oprette skemaerne i ”onsucces”, ved at køre to for-løkker der tager alle elementer fra resultatet fra serveren og tjekker dem mod de skemaer vi har oprettet for at tjekke om der allerede er et skema, som de skal sætters på eller om det er det første element, så der skal oprettes et nyt skema. Vi tjekker også for om der overhovedet er undervisning for det pågældende kursus den uge ved at tjekke mod den uge der er valgt i vores ugetæller.
Hvis der skal oprettes et nyt skema sker det ved at der bliver oprettet en ny flextable for det valgte lokale som bliver tilføjet til listen som vi tjekker mod, og tilføjet til det panel som sørger for, at det bliver vist på siden.

Hvis der skal oprettes et nyt skema sker dette med ”createScheme” funktionen. Den opretter et skema med ugedagene og tidspunkter, hvor alle skemagrupper er frie. I øverste venstre hjørne bliver lokalenavnet sat.

Herefter bruger vi funktionen ”addToScheme” til at finde ud af hvor i vores skema et kursus skal sættes ind. Først finder vi ud af hvilken ugedag kurset ligger, så kigger vi på hvad tid på dagen kurset starter, og til sidst tjekker vi hvor mange halvtimer det løber over. Med disse informationer ved vi hvilke celler i skemaet de skal sættes ind på og om de strækker sig over mere end en celle. Der er også et tjek for om vi er i den sidste uge, da dette skema er lidt anderledes.
Det sidste vi tjekker er om vi er i standardview eller adminview dette bruges til at bestemme om vi skal skrive dobbeltbookninger ind og om vi skal have kommentarer eller billeder ved siden af.

Konklusion

Vi har gennem projektet udviklet en prototype til et værktøj, hvis formål det er at skabe bedre overblik i forhold til skemalægning på DTU ud fra forelæsernes ønsker.

Vi har arbejdet med to visninger, hvor den ene er tænkt som et administrativt værktøj, der giver et enkelt overblik over hvilket ændringer der skal foretages for at skemalægningen går op.

Den anden er et bud på et værktøj, til dem der skal bruge skemaerne i det daglige, og giver et overblik over hvilke lokaler der er optaget, af hvilke fag, og ikke mindst hvor på campus lokalerne er beliggende. Denne visning er også tænkt som et forslag til hvordan det ser ud efter skemalægningen er overstået, dvs. når alle kurser har et lokal og der ikke er flere dobbeltbookninger tilbage. Dobbeltbookninger fjernes ikke automatisk da dette ville kræve en intelligent agent, der kan tage højde for afstande til andre bygninger og kommentarer, og dette har ikke været formålet med denne opgave.

Forslaget til videreudvikling med bookning på siden, som vi har foreslået ville dog bringe os et stykke af vejen.
Vi er kommet godt rundt om problemet ud fra de forudsætninger vi havde til projektet, men det kunne være spændende at arbejde videre med projektet, i forhold til at lave et mere automatiseret system.

Litteraturliste

Hjemmesider

- http://www.gwtproject.org/
- Mysql.com

Videoer/guides

- https://www.youtube.com/playlist?list=PLoWne5q-c9E92-rrra5eiztpq1ACrCWNm

Vi har brugt alle videoerne der finde på den playlist der er linket til under videoer.