Technical University of Denmark

Brugerinterface til Ekstern Hardwarekommunikation

Course:
Bachelor Projekt
Institut for Matematik og Computer Science

Author: Jens Grossmann Larsen (s135254)

Dato: Onsdag, 8. Maj - 2016

1 Abstract

In the future every thing will have the ability to go online and be controlled over the internet. It is therefore vital that we have the right tools to make systems that are easy to handle through the cloud. This report contains the creation of a system to control and operate an industry weight through an userinterface made with Google Web Toolkit and includes the strengths and weaknesses of the development environment.

I fremtiden vil alt være i stand til at gå online og blive styret over internetet. Det er derfor vigtigt vi finder de bedste værktøjer til at kunne lave systemer som nemt kan håndteres igennem skyen. Denne raport indeholder opbygningen af et system til at kontrollere og styre en industri vægt over nettet igennem en brugergrænseflade, ved hjælp af Google Web Toolkit, samt de styrker og svagheder som udviklingsmiljøet medbringer.

2 Distribution of Work

Dette projekt var lavet i fællesskab mellem Jens Grossmann Larsen og Frederik Haaning.
Denne rapport er skrevet af Jens Grossmann Larsen med fokus på Windows og Eclipse. En tilsvarende rapport kan findes af Frederik Haaning med fokus på Linux og IntelliJ
page

Table of Contents

3 Indledning

Meningen ved dette projekt er at prøve at få en brugergrænseflade på en hjemmeside til at styre en industri vægt. Dette gøres ved hjælp af googles GWT, som er et udviklingsmiljø til at lave store projekter der kræver trafik i begge retninger. Rapporten har fokus på Windows versionen, samt Eclipse som IDE til at håndtere GWT. Projektet kunne omhandle alle former for hardware og hårde hvide varer, men en vægt er særlig interersant da den skal have sendt konstante opdateringer igennem nettet.

4 Specifikationer

Til følgende projekt bruger vi følgende Hardware udstyr:
- Mettler Toledo Industrivægt
- Huawei Mobil-netværks router
- Windows laptop (Denne rapport)
- Linux Laptop (Frederiks rapport)
- Raspberry Pi
- Kabler (ethernet, strøm, etc.)

Til følgende projekt er brugt følgende software:
- Eclipse (Denne raport)
- IntelliJ (Frederiks rapport)
- valgfri Browser
- Putty
- Tomcat
- Git

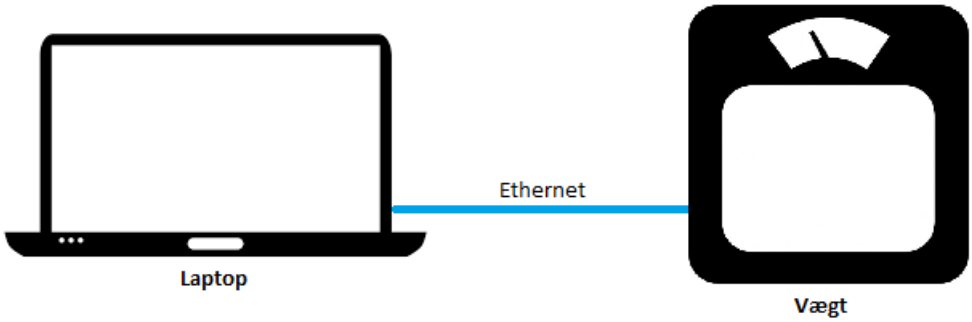
5 Industrivægten

Inden der blev arbejdet med GWT, skulle der først læres hvordan vægten fungerede. En Mettler Toledo vægt blev taget i brug

5.1 Direkte forbindelse til vægten

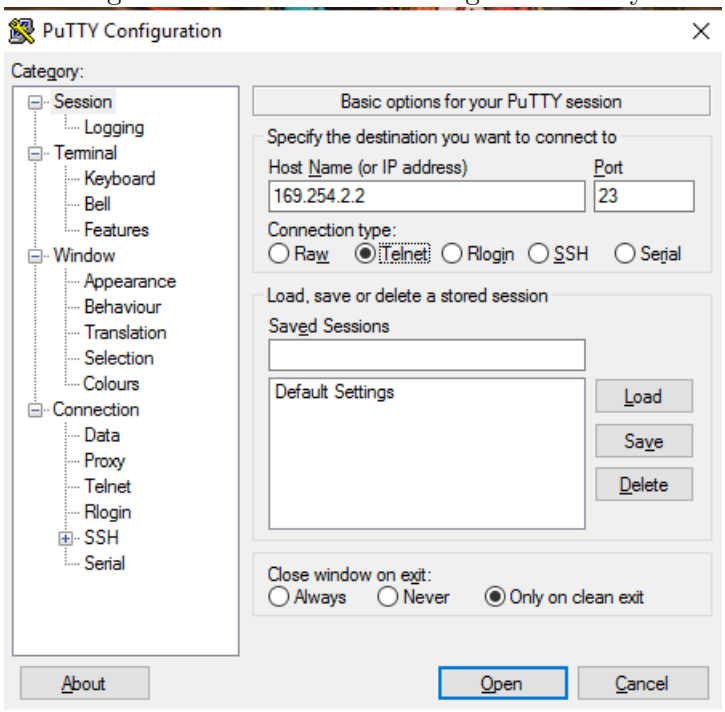
Det første vi ville gøre var at lære vægten at kende. Derfor ignorerede vi brugen af Huawei routeren og GWT, og opsatte bare en direkte forbindelse mellem en laptop og vægten med et ethernet kabel.

Figure 1: Setup vi brugte i starten til at arbejde med vægten



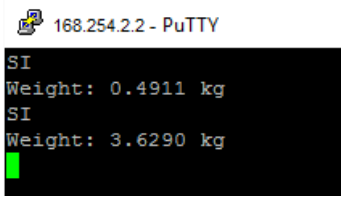
Til at starte med brugte vi den helt standard command-prompt (CMD). Når man brugte CMDs ping kommando med vægtens IP og port, fik vi ping resultater, som fortalte os der var hul igennem. Næste skridt var at give vægten kommandoer som den så ville give respons på (f.eks. nuværende vægt). Vægten ville dog ikke adlyde kommandoerne når den modtog dem gennem CMD uanset hvad der blev gjort. Vi Prøvede at køre igennem en telnet forbindelse eller at ændre vores egen IP/Port, men uden held

Figure 2: Direkte forbindelse igennem Putty



Da rendte vi ind i et lille program ved navn Putty. Putty havde et simpel ui, som bare bad om IP'en og porten, begge som vi havde. Putty åbner da et lille CMD-lignende interface hvor vi kunne skrive kommandoer til vægten, men til forskel for den normalle CMD, gav vægten respons til vores kommandoer som forventede. Vi kunne nu afprøve alle vægtens kommandoer for at finde ud af hvilke vi skulle bruge.

Figure 3: Putty får svar fra vægten



5.2 Vægtens hovedkommandoer

Mettler vægten inkluderede to store dokumentations dokumenter, hvor den ene kun indeholdte de mange kommandoer som vægten kunne modtage. Alle kommandoer blev afprøvet, nogen med mere succes end andre. Vi endte med kun at bruge en meget lille brøkdel af vægtens muligheder, da simplicitet var vigtigt for os. Igennem projektet historie har kommandoerne lige så stille erstattet hinanden med mere eller mindre komplekse kommandoer afhængig af kodens struktur. I den sidste iteration af koden brugte vi kun SI og TI.

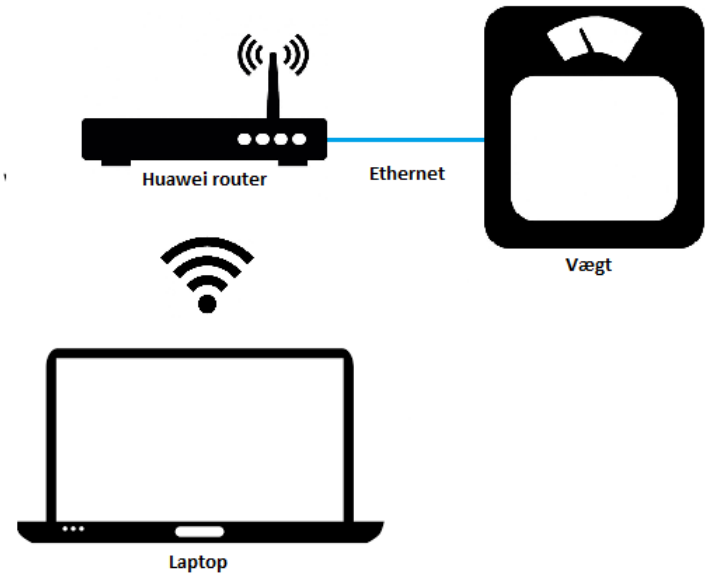
Table 1: Tabel over kommandoerne vi brugte

Kommando	**Funktion**	**Hvad blev den brugt til i projektet**
D "tekst"	Erstatter displayet på vægten med hvad der står i "tekst"	Den blev brugt meget i starten, inden vi gik i gang med GWT, til at teste forbindelsen mellem laptop og vægt.
S	Returner vægten når den har stabiliseret	Blev brugt til at modtage vægten, men virkede ikke ordentligt da vi konstant skulle opdatere displayet selvom vægten ikke var stabil
SI	Returner vægten uanset stabilitet	Det vi endte med at bruge til at få konstant vægt fra vægten, da den ignorer stabilitet.
SIR	Returnerer vægten uanset stabilitet 7-8 gange i sekundet	Da vægten konstant skulle opdatere virkede den som det rigtige valg, men det fungerede bare ikke ordenligt i vores logik
T	Tare vægten når den er stabil	Brugt i starten til Tare-delen af hjemmesiden
TI	Tare vægten uanset stabilitet	Tare delen var til tider for langsom da den ventede på stabil vægt, så vi brugte den her i stedet
PWR 0/1	Slukker eller tænder vægten afhængig om det er 1 eller 0 efter PWR	Planen var at bruge den kommando til at slukke og tænde vægten, men vi fandt aldrig rigtig tid til at implementere den

6 Huawei - Mobilrouter

Efter at have forbundet direkte til vægten var det tid til næste skridt. Der blev nu inkluderet Huawei's mobilnetværks router. Vi forbandt vægten til routeren igennem et ethernet stik og forbandt laptop og router gennem wifi.

Figure 4: Det setup vi arbejde med det meste at projektet

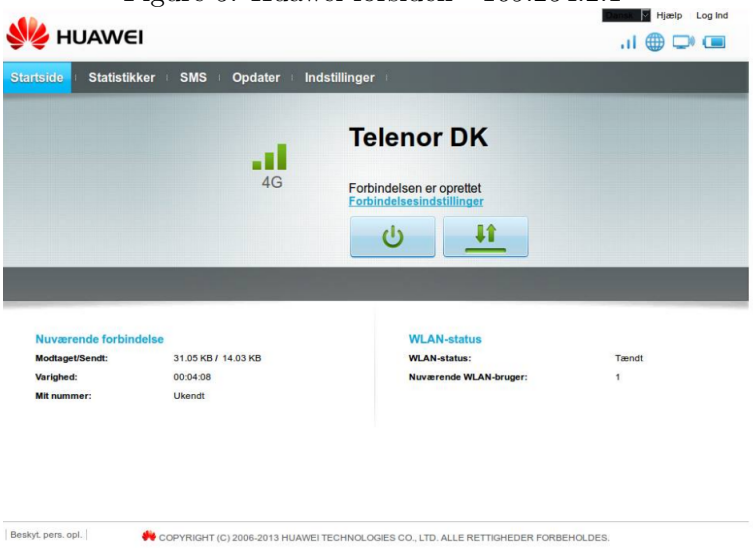


Det var dog mere kompliceret end bare at forbinde de tre stykker hardware. Nu skulle vi ind og ændre på Huawei'ens router indstillinger.

6.1 Opsætning af Huawei

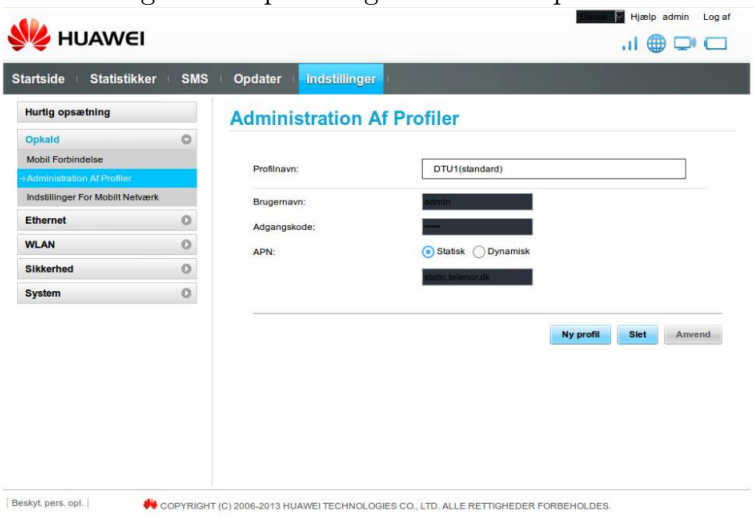
Huawei'en kørte på ip-adressen 169.254.2.1, og ved at indtaste det ind i en browser kom vi frem til routers egen webpage.

Figure 5: Huawei forsiden - 169.254.2.1



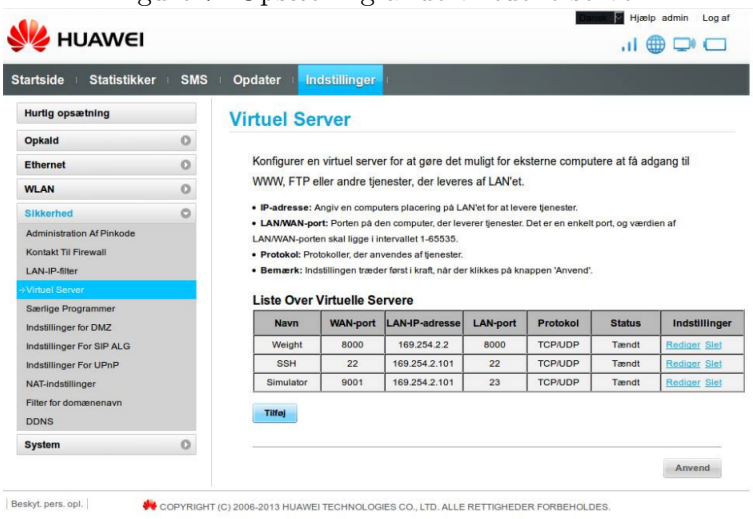
Login oplysningerne stod på huawei'en så det var nemt at logge ind. Første skridt var at lave en profil til DTU med en statisk IP, som DTU har købt af telenor.

Figure 6: Opsætningen af Huawei profiler



Så manglede der bare alle de virtuelle serverer. Når Huawei'en modtager en forbindelse der er rette mod en bestemt port videresender den dig til den angivne IP. Igennem projektet har vi haft mange serverer oppe, men til sidst havde vi kun brug for 3. Vægten er den vigtige her, da vi skulle kunne forbinde med vægten igennem huawei'en og simulatoren er kun blevet brugt til testing, som nævnes i senere afsnit.

Figure 7: Opsætning af de virtuelle server



7 Hvad er GWT?

GWT (udtales officielt "Gwit", men alle kalder det "GVT") står for "Google Web Toolkit" og er et gratis udviklingsmiljø udviklet af Google, som senere hen er blevet open source. Formålet med GWT er at kunne lave complekse hjemmesider og applikationer uden større forståelse af kodningssprog som Javascript(JS). Dette gøres ved at GWT cross-compiler helt normal Java kode (med undtagelse af nogen specielle plugins) om til Javascript og du har derfor alle fordelene ved Java som Javascript. Mange skal ofte vælge om de vil bruge JS eller GWT og bliver tit besluttet på størrelsen af projektet, da JS er rigtig godt til små projektet, men knap så godt til større. Man kommer hurtigt igang med JS, og får skrevet en masse kode, men problemet opstår når koden bliver for stor og man skal debug. JS mangler virkelig mange tools til debugging og det hjælper hellere ikke at mange IDEs ikke understøtter JS til sit fuldeste.

GWT har der imod mange gode tools selvom om projektet er blevet stort. GWT genereret JS kode er også meget mere kompakt end de fleste programmører ville kunne formå, hvilket også giver meget bedre struktur over koden. GWT har også den fantastiske evne til at fjerne overskydende bibloteker som ikke bliver brugt.

De fleste sprog og programerings miljøer loader alle pakkerne ved hver load, men GWT loader kun dem der lige præcis er brug for hvilket giver effektivitet i sidste ende.

HTML er dog meget nemmere at arbejde med når det ikke skal bruges til GWT. HTML kode er i sig selv meget simpelt at kode begrund af HTMLs simple koncept. Men når man skal implementere det igennem Java kode i GWT, bliver det pludseligt mere kringlet og uoverskueligt.

Dog er en af GWTs største styrker den måde den sammensætter backend og frontend. Dette er meget besværligt hvis man bare kører ren HTML, CSS og Javascript kode, da Javascript er enkelt trådet og derfor ikke kan lave en ny forespørgselse før den får svar fra den sidste. GWT løser det med det der hedder AJAX programmering.

7.1 AJAX

AJAX står for ***Asynchronous JavaScript and XML*** og er en gruppe af udviklinger som har til formål at løse forespørgelses problemet der opstår ved traditionelle hjemmesider. AJAX gør at man kan sende og modtage data fra serverer asynkront, hvilket gør at hjemmesiden ikke bliver ændret eller stoppet ved hver forespørgelse hjemmesiden laver.

AJAX har et par kerne programmer, som alle er forbundet igennem JavaScript:

HTML(HyperText Markup Language) og **CSS(Cascading Style Sheets)** er basen for næsten alle hjemmesider og producerer det look som hjemmesiden ender med.

Derudover har AJAX det der hedder **DOM (Document Object Model)**, som opbyger et DOM-træ af hjemmesiden, som andre dele af hjemmesidens kan hente og ændre/redigere på.

JSON(JavaScript Object Notation) og **XML(Extensible Markup Language)** er begge brugt til at lave nogen reglsæt som kan bruges til at inkode et dokument, så det er læsligt af både mennesker og maskiner. XML er et markup-language som HTML, hvor JSON er en måde at repræsentere elementer på.
XML er dog den AJAX bruger mest, hvor JSON er mere brugt til AJAJ(asynchronous JavaScript and JSON), hvilket også bliver afsløret af deres navnen.

```
<Document>
    <Paragraph Align="Center">
        Here <Bold>is</Bold> some text.
    </Paragraph>
</Document>
```

Figure 8: XML Eksempel

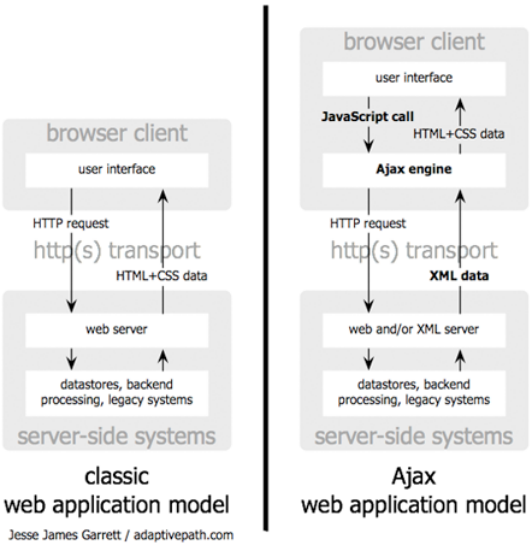
```
{
    "Paragraphs": [
        {
            "align": "center",
            "content": [
                "Here ", {
                    "style" : "bold",
                    "content": [ "is" ]
                },
                " some text."
            ]
        }
    ]
}
```

Figure 9: JSON Eksempel

Til sidst er der også **XHR(XMLHttpRequest)** som er en API som Javascript kan bruge til at sende HTTP og HTTPS anmodninger og bruge svaret fra webserveren i sin kode. GWT gør god brug af denne del af AJAX til at sende objekter og filer i Java.

AJAX bliver også brugt i GWTs **RPC(Remote procedure call)** som gør det muligt for data overførelse for GWTs applicationer. RPC'en gør også at Java koden bliver pakket ind i AJAX, som gør at den ikke længere går op i hvilken browser type man bruger.

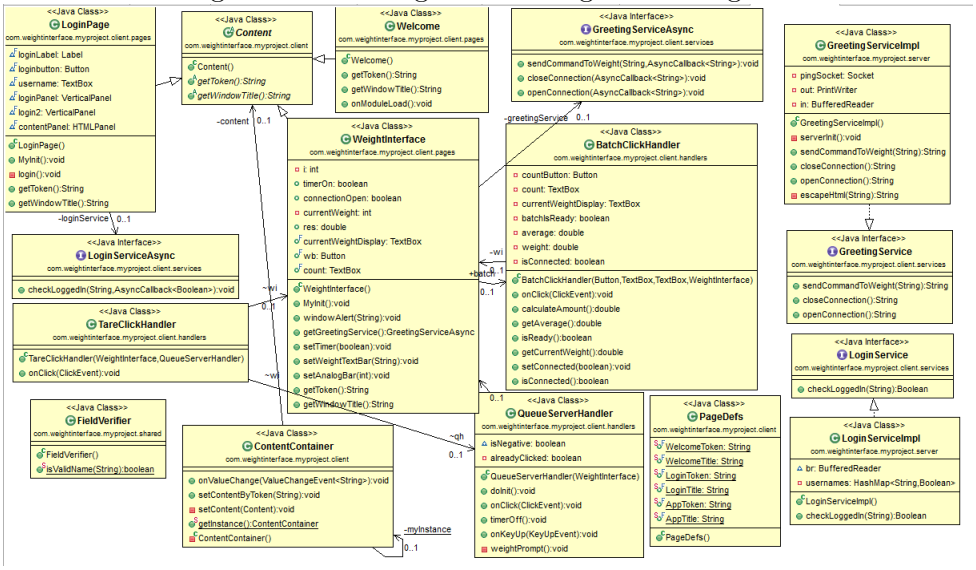
Figure 10: AJAX



8 Implementering

Da der er rigtigt mange klasser får man et enormt klassediagram hvis man inkluderer alle klassernes metoder. Derfor forklares koden istedet i mindre dele for at gørre det mere overskueligt.

Figure 11: Stort og uoverskueligt klassediagram



8.1 Frontend

Frontend delen af koden er den som brugeren interagerer med. Frontenden består af både interaktive og ikke-interaktive elementer.

8.1.1 HTML

HTML eller *HyperText Markup Language* er et markup language (som navnet indikere). HTML er simpelt i sin brug, men ikke særlig brugbart i sig selv. HTML laver alle elementerne på en hjemmeside, som tekst, overskrifter og de usynlige kasser de befinder sig i for at holde styr på dem.

I vores GWT kode implementere vi HTML ved at lave nye *VerticalPanel, HorizontalPanel og HTMLPanel*. Vertical- og HorizontalPanel er ikke særlig fleksible, men gode til det grove arbejde. HTMLPanel er mere fleksibelt og kan introducere alle former for HTML elementer til koden. Det er også her vi introducere GWT widgets som knapper og textfields.
Denne form for HTML er dog ikke standard metoden at implementere HTML med, men en nødvendighed når man arbejder med GWT. Normal HTML har en mere logisk

strukturering end HTML implementeret gennem Java. For at kunne lave HTML i Java kode skal der huskes at hente pakken *import com.google.gwt.user.client.ui.*;*. Desværre er ren HTML kode både grimt og ufleksibelt. Derfor bruger man et Style sheet som CSS til at pifte koden lidt op.

Figure 12: Eksempel på HTML kode igennem GWT

```
//Weight
weightPanel.add(new HTMLPanel("\t\t<h1>Weight Control</h1>\n" +
        "\t\t<a href=#><div class=\"tothetop\">Top <i class=\"fa fa-chevron-up  color-blue \"></i></div></a>\n"));
weightPanel.add(analogBar);
weightPanel.add(weightPanelControls);
weightPanelControls.add(wb);
weightPanelControls.add(currentWeightDisplay);
weightPanelControls.add(tareButton);
```

8.1.2 CSS

CSS eller *Cascading Style Sheets* er kode der fortæller HTML hvordan det skal se ud. Dette gøres ved at kalde HTML elementerne direkte eller give HTML koden klasser og id'er.
For de horizontale og vertical paneler giver vi dem deres CSS direkte ved funktioner som *panel.setWidth("100%")*. Alle vores HTMLPaneler har derimod alle blevet givet enten en klasse eller en id, som vi så refererer til i vores CSS dokument. For at give et element en klasse bruges .addStyleName("navn"). Dette gør at vi kan gruppere vores elementer, så vi kan give flere ting styling på en gang.

Figure 13: Eksempel på standard CSS kode

```
.weightDisplay {
    border: 0 solid;
    border-radius: 10px;
    font-size: 2em;
    width: 110%;    color: #fff;
    background-color: #ddd;
    text-shadow: 0.035em 0.035em #000;

}

.tareButton {
    border: 0 solid;
    border-radius: 10px;
    font-size: 2em;
    color: #fff;
    background-color: #00A9DF;
}
```

8.2 Backend

Backend er den del som brugeren ikke oplever. Det der sker når man klikker på en knap eller indtaster et username i et textfield. Det vil sige at backend er den del som gør at frontend giver mening for brugeren istedet for bare at være pæne former på en skærm.

8.2.1 ContentContainer og Container

I de sidste par itterationer af vores projekt havde vi mere end en page (WeightInterfacePage og LoginPage). Derfor måtte vi implementere en måde hvor på vi nemt kunne udskifte HTML elementerne med nogen nye. Vi bruger klassen ContentContainer.java og den abstracte klasse Content til at holde styr på hvilken HTML side den skal bruge.

Nogen af de interactive elementer i HTML kalder setContentByToken(token) med en variable korrendsponderende til enten LoginPage eller WeightInterface. Afhængigt af hvilken token setContentByToken(token) får, sender den en abstrakt klasse til setContent(content) med hvilken HTML-side den skal vise.

Figure 14: setContentByToken

```
public void setContentByToken(String token) {
    if (token.equals(PageDefs.WelcomeToken)) {
        this.setContent(new Welcome());
    } else if (token.equals(PageDefs.AppToken)) {
        this.setContent(new WeightInterface());
    } else if(token.equals(PageDefs.LoginToken)) {
            this.setContent(new LoginPage());
    }
}
```

setContent(content) sletter derefter alle nuværende HTML-elementer og tilføjer alle den nye HTML kode afhængigt af hvilken Content den modtog.

Figure 15: setContent

```
private void setContent(Content content) {
    RootPanel contentRoot = RootPanel.get("content");
    contentRoot.clear();
    this.content = content;
    History.newItem(content.getToken(), false);
    // check for special initializations:
    if (content.getToken().equals(PageDefs.AppToken)) {
        ((WeightInterface) content).MyInit();
    }
    if (content.getToken().equals(PageDefs.LoginToken)) {
        ((LoginPage) content).MyInit();
    }
    contentRoot.add(content);
    Window.setTitle(content.getWindowTitle());
    Window.scrollTo(0, 0);
}
```

Welcome.java er en lille klasse som kun blev brugt til at debugge, da vi kan vælge om den skal starte med LoginPage eller WeightInterface, hvilket gjorde vi ikke skulle logge ind efter hver lille ændring vi havde lavet.
PageDefs.java bruges bare til de id'er som ContentContainer bruger for at se hvilken webpage den skal opsætte, et slags look-up table.

```
public void onModuleLoad() {
    new LoginPage();
    //new WeightInterface();
}
```

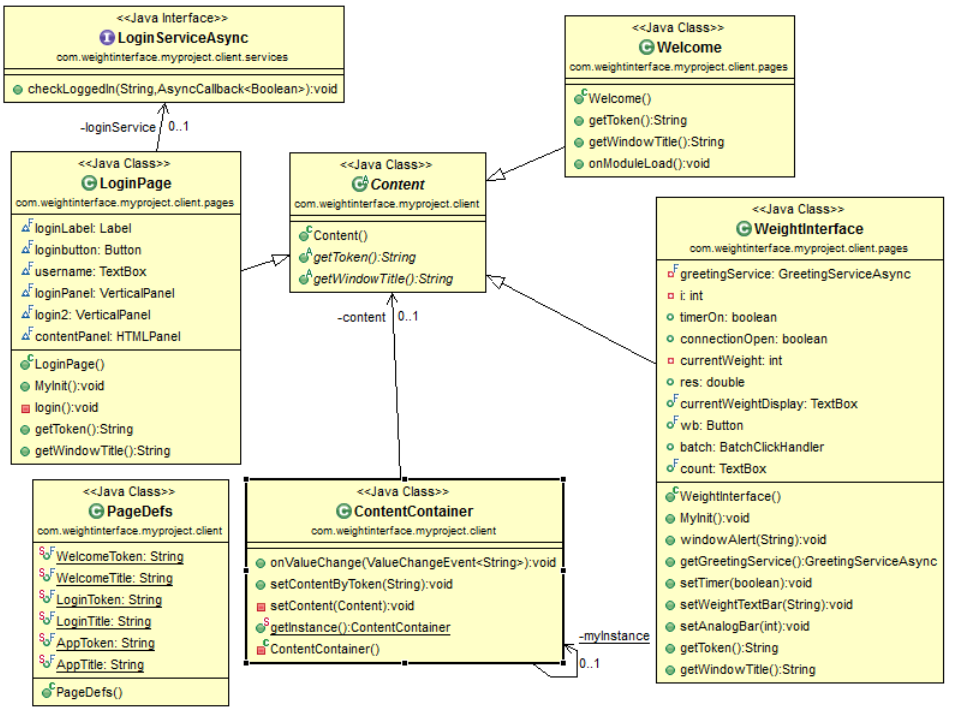
Figure 16: Welcome.java beslutter startside

```
public class PageDefs{
    public static final String WelcomeToken = "WELCOME";
    public static final String WelcomeTitle = "Welcome";
    public static final String LoginToken = "LOGIN";
    public static final String LoginTitle = "Login";
    public static final String AppToken = "WI";
    public static final String AppTitle = "WeightInterace";
}
```

Figure 17: PageDefs.java

Det giver os følgende klasse diagram for content delen af projektet:

Figure 18: Klassediagram over Content håndtering



8.2.2 LoginService

Ambitionsniveautet var højt da vi besluttede at lave en login side ved hjælp af SQL som ingen af os havde arbejdet med før. Det endte ikke med en SQL server, men i stedet en simpel, men flot, login side som bare checkede om dit brugernavn var korrekt ved hjælp af Hashmap.

Figure 19: LoginService checker om username er korrekt

```
public Boolean checkLoggedIn(String username) throws IOException {
    URL path = LoginServiceImpl.class.getResource("usernames.txt");
    File f = new File(path.getFile());
    br = new BufferedReader(new FileReader(f));
    usernames.clear();
    String line = null;
    try {
        line = br.readLine();
    } catch (IOException e) {
        e.printStackTrace();
    }

    while(line != null)
    {
        usernames.put(line,Boolean.TRUE);
        try {
            line = br.readLine();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    if (usernames.get(username).booleanValue()) return Boolean.TRUE;
    else return Boolean.FALSE;
}
```

Koden checker da om det indtastet brugernavn er med i listen af korrekte brugernavne (eller om der overhovedet er indtastet en brugernavn). afhængigt om det stemmer overens returnerer den sand eller falsk tilbage til LoginPage.
Hvis LoginPage får et True returneret, fortæller den ContentContainer at den skal skifte til WeightInterface.

8.2.3 GWTs RPC

Figure 20: GWTs RPC pakke

```
import com.google.gwt.user.client.rpc.RemoteService;
```

GWT har sin egen RPC klasse, som er specifikt lavet til GWT. Client og server delene af programet skal extend RemoteService eller RemoteServiceServlet delen af GWTs RPC pakke.

Figure 21: Implementation af RPC for kommunikation med vægt

```
public interface GreetingService extends RemoteService {
  //String greetServer(String name) throws IllegalArgumentException;
  String sendCommandToWeight(String command) throws IOException;
  String closeConnection() throws IOException;
  String openConnection() throws IOException;
}
```

Som det ses skal alt der kommunkere med vægten bruge RPC'en: sendCommandToWeight (sender kommando), closeConnection (slukker forbindelsen) og openConnection (starter forbindelsen).

Figure 22: aSync Callback using RPC

```
public interface GreetingServiceAsync {

  //String sendCommandToWeight(String s) throws IOException;
  void sendCommandToWeight(String command, AsyncCallback<String> callback)
      throws IOException;

  void closeConnection(AsyncCallback<String> asyncCallback) throws IOException;
  void openConnection(AsyncCallback<String> asyncCallback) throws IOException;

}
```

8.2.4 GreetingService

GreetingsService er linjen mellem resten af koden og vægten. Den starter med at lave et socket til at forbinde til vægten. Vi har også andre sockets klar her i tilfældet af vi vil skrive med vores simulator eller Frederiks stationære.

Figure 23: Opretter et socket

```
private void serverInit() throws IOException {
    //pingSocket = new Socket("192.168.1.214", 23);
    //pingSocket = new Socket("127.0.0.1",23);
    pingSocket = new Socket("62.79.16.16",8000);
    out = new PrintWriter(pingSocket.getOutputStream(), true);
    in = new BufferedReader(new InputStreamReader(pingSocket.getInputStream()));
}
```

Nu kan de andre klasser kalde sendCommandToWeight(command) igennem GreetingsService for at sende kommandoer til vægten, som så returnerer et svar til kommandoen (f.eks. nuværende vægt).

Figure 24: sendCommandtoWeight

```
public String sendCommandToWeight(String command) throws IOException {
    String result = null;
    out.println(command);
    result = in.readLine();
    return result;    }
```

8.2.5 Vægt - QueueServerHandler

Nu hvor vi har oprettet en måde at sende kommandoer til vægten kan vi lave et loop til at håndtere den konstante opdatering af vægten, som vi gerne vil ha for at det føles som et rigtig vægt display.

weightPrompt, som automatisk bliver kaldt, sender et "SI" gennem sendCommand-ToWeight. Dette får vægten til at returnere nuværende vægt tilbage til weightPrompt.

Figure 25: weightPrompt sender "SI" til vægten

```
private void weightPrompt() throws IOException {
    wi.getGreetingService().sendCommandToWeight("SI", new AsyncCallback<String>() {
        public void onFailure(Throwable caught) {
```

weightPrompt har derefter en onFailure og onSucces metode. Når den har succes, tjekker den om værdien er negativ eller om batch er aktivt. Uanset hvad opdaterer den vores HTML element med den nye vægt. Til sidst kalder den sig selv sår vi kan få et loop op og kører, så den kan opdatere vægten så hurtigt som programmet og vægten kan sende mellem hinanden. Dette gør også at der ikke "hobber" sig kommandoer op i vægten, da weightPrompt først sender en ny kommando når den gamle er modtaget og bearbejdet.

Figure 26: weightPrompt onSucces håndtering

```
public void onSuccess(String weight) {
    //DOM.getElementById("sk-circle").getStyle().setDisplay(Style.Display.NONE);
    isNegative = false;
    wi.wb.setText("OFF");
    char c = weight.charAt(7);
    if (!(c == ' ')) isNegative = true;
    wi.res = Double.parseDouble(weight.substring(8,14));
    if(wi.batch.isReady()){
        double batchdouble = wi.batch.calculateAmount();
        wi.count.setText(""+(int) batchdouble);};
    if(isNegative)
    {
        wi.currentWeightDisplay.setText("-"+wi.res+" kg");
        DOM.getElementById("myBar").getStyle().setWidth(0, Style.Unit.PCT);
        DOM.getElementById("label").setInnerText("0%");
    } else
    {
        wi.currentWeightDisplay.setText(wi.res + " kg");
        DOM.getElementById("myBar").getStyle().setWidth(wi.res * 16.66, Style.Unit.PCT);
        DOM.getElementById("label").setInnerText(Double.toString(wi.res * 16.66).substring(0, 6) + "%");
    }
    try {
        weightPrompt();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

For at sikre os at QueueServerHandler ikke skal lave nye instancer af klassen hver gang den kalder weightPrompt, sættes weightInterface variablen wi til null, som så bliver gennemoprettet ved hvert kald.

Figure 27: håndtering af weightInterface i QueueServerHandler

```
WeightInterface wi = null;
boolean isNegative = false;
private boolean alreadyClicked = false;

public QueueServerHandler(WeightInterface wi)
{
this.wi = wi;
    wi.wb.setVisible(false);
    doInit();
}

public void doInit() {
    try {
        weightPrompt();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

8.3 Vægt-Simulator

Mettler Teledo vægten blev brugt til undervisning mens vi arbejdede på projektet, hvilket gjorde vi ikke bare kunne tage den med hjem. Det var klodset altid at skulle hen til DTU for at teste features som brugte vægten. Dette fik os til at lave en simulator af vægten, så man kunne teste vægt specifikke features uden at hente den.

Figure 28: Vigtiste kode i Simulatoren

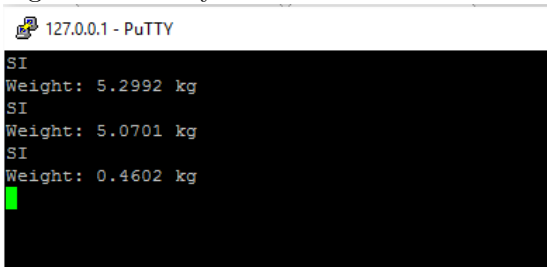
```
try
{
    PrintWriter out2 = new PrintWriter(getOutputStream(),true);


    Random random = new Random();
    BufferedReader in =
            new BufferedReader(
                    new InputStreamReader(clientSocket.getInputStream()));
    String line;
    while((line = in.readLine()) != null)
    {
        System.out.println(line);
        if(line.contains("SI"))
        {
            out2.println("Weight: "+getRandomValue(random,0,6,4)+" kg");
        }
        else if(line.contains("TI"))
        {
            out2.println("Tare");
        }
    }


}
```

Simulatoren var opbygget af simpel Java kode og havde bare til formål at sende en tilfældig gyldig værdi tilbage når den modtog *SI* kommandoen som normalt sender nuværende vægt tilbage. Det vigtiste var at det den sendte retur var formateret rigtigt, som antallet af decimaler, mellemrum og ordet "kg".

Figure 29: Putty forbundet til Simulatoren



9 Installation af GWT i Eclipse

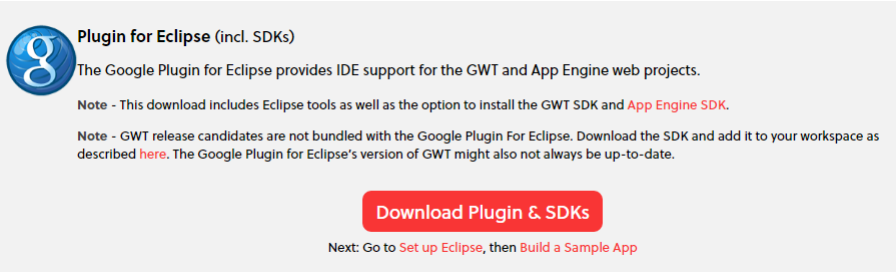
GWT kan installeres på de fleste store operativsystemer og et stort udvalg af udviklingsmiljøer (IDE - Integrated Development Enviroment). Dette gjorde det nemt da projekt gruppen bestod af en windows og en linux bruger. Gruppen installerede derfor GWT hver for sig og endte med at bruge to forskellige udviklingsmiljøer, "Eclipse" på Windows og "IntelliJ" på Linux.
Følgende afsnit omhandler installationen af GWT i Eclipse på Windows. For IntelliJ installation henvises der til Frederiks rapport som inkludere GWT i IntelliJ på et Linux operativsystem.

Advarelse: Læs hele guiden, problemerne og begge metoder før du påbegynder installationen, da tingene har ændret sig siden.

9.1 GWT Plugin

Den første metode vi brugte til at installere GWT i eclipse var ved at downloade og installere deres GWT Plugin for eclipse.

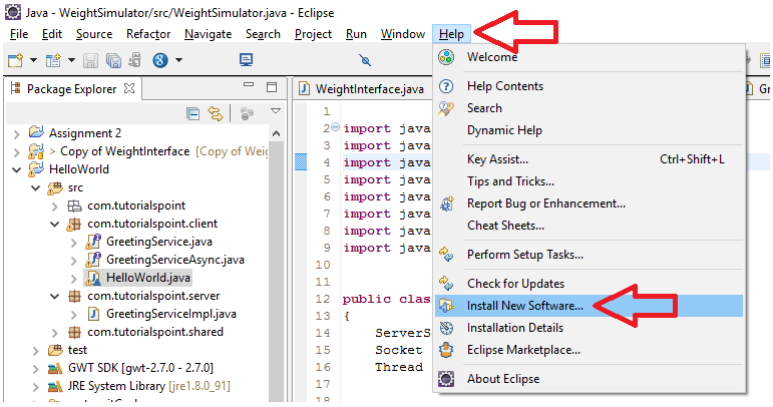
1) Gå ind på download sectionen af GWT's hjemmeside og Klik på "Download Plugin & SDK"



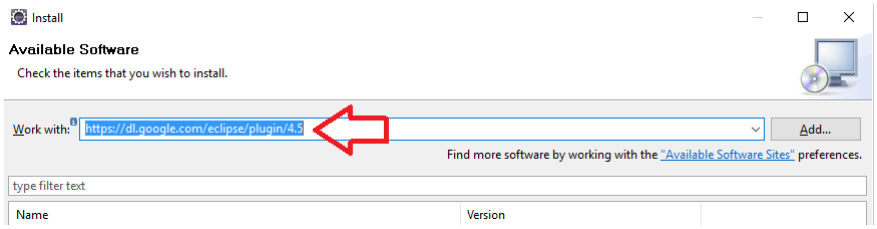
2) Vælg den version af Eclipse som du har installeret (Vores projekt brugte 4.5 Mars) og kopier (eller husk) det tilsvarende link under "Direct plugin link".

Eclipse version	Installation instructions	Direct plugin link
Eclipse 4.5 (Mars)	Plugin for Eclipse 4.5 (Mars)	https://dl.google.com/eclipse/plugin/4.5
Eclipse 4.4 (Luna)	Plugin for Eclipse 4.4 (Luna)	https://dl.google.com/eclipse/plugin/4.4

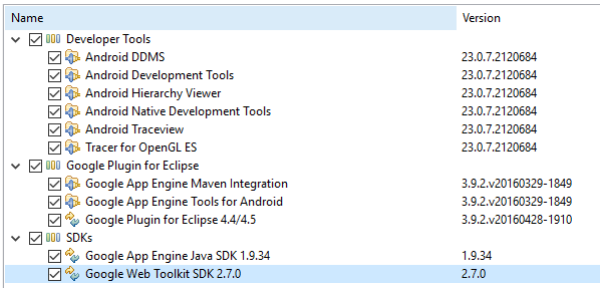
3) Åben Eclipse og vælg "Help" → "Install new software..."



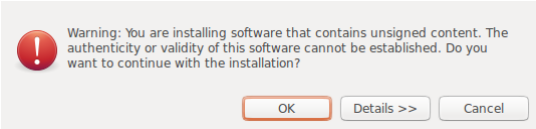
4) indsæt linket fra trin 2 i feltet "Work with"



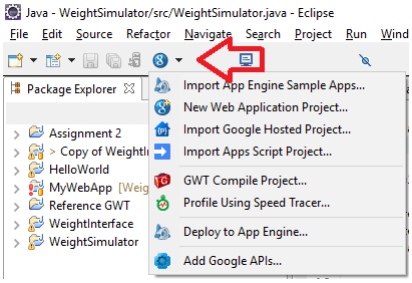
5) Vælg hvilke dele af GWT du vil installere. Kan klikke på dem for flere oplysninger, men vi installerede bare alle for en sikkerheds skyld. Noter at den siger 2.7.0 under SDK (Nyeste version).



6) Derefter kommer der oplysninger om hvad du er ved at installere og license agreements. Hvis der kommer en Sikkerheds advarelse, bare klik "Ok" og derefter beder den om at genstarte Eclipse.



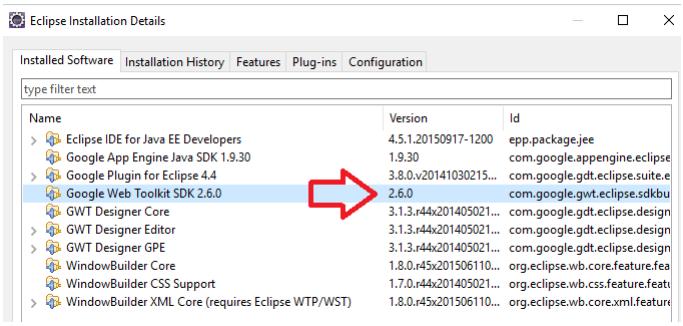
7) Du har nu GWT, installeret på Eclipse.



9.1.1 Problemer efter installationen af plugin

Der opstod hurtigt problemer med pluginet fra deres hjemmeside. Mine egne GWT programer, som "Hello World", virkede som de skulle, men der var et problem med at hver gang man lavede en ændring i koden skulle man compilere igen, som tog hele idéen bag superdev mode væk.

Andet problem var at den ikke kunne køre de GWT filer som jeg modtog fra min partner. Det viste sig at SDK 2.7.0 af en eller anden grund installerede 2.6.0 SDK'en som ikke ville køre 2.7.0 filerne korrekt. Den anden metode vi brugte endte med at løse problemet.

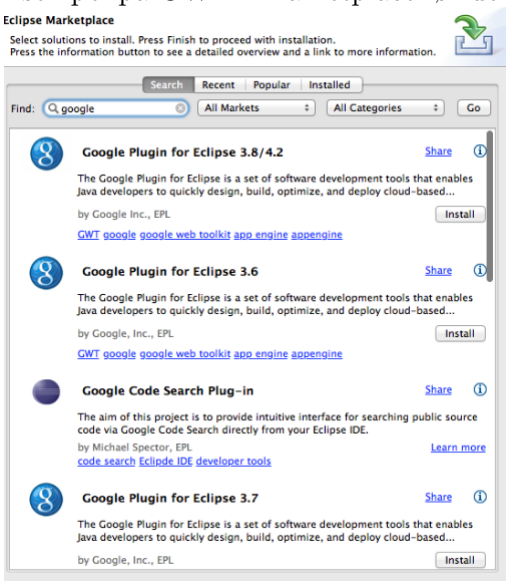


9.2 Løsning til problemerne - Eclipse Marketplace

Den anden metode, som endte med at fixe plugin problemer, var at gå ind på Eclipse Marketplace og søge på GWT. En 2.6.0 og en 2.7.0 version kommer frem og 2.6.0 delen stod allerede som installeret. Installerede 2.7.0 versionen og det hele kørte som det skulle.

VIGTIGT! Siden vi installerede GWT, ser det ud til at GWT er blevet fjernet fra Eclipse Marketplace. Den plejede at vise 2.6.0 og 2.7.0 versioner når man søgte på GWT, og begge versioner er fjernet fra min "installeret" del af market (havde dem begge). Det vil sige jeg er tilbage på 2.6.0, men har ingen af de problemer vi originalt havde med 2.6.0.

Figure 30: Eksempel på GWT i Marketplace før de fjernede den

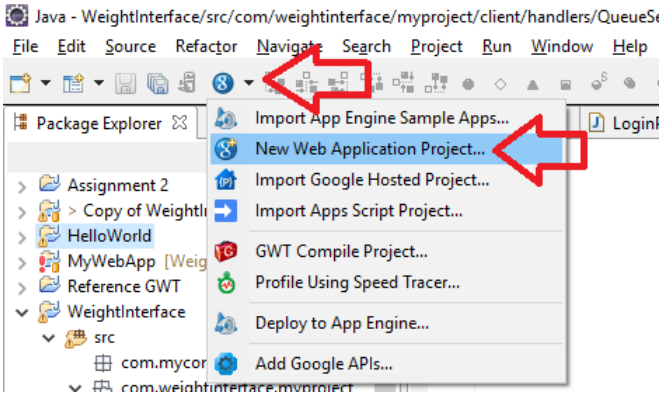


10 Hvordan man bruger GWT

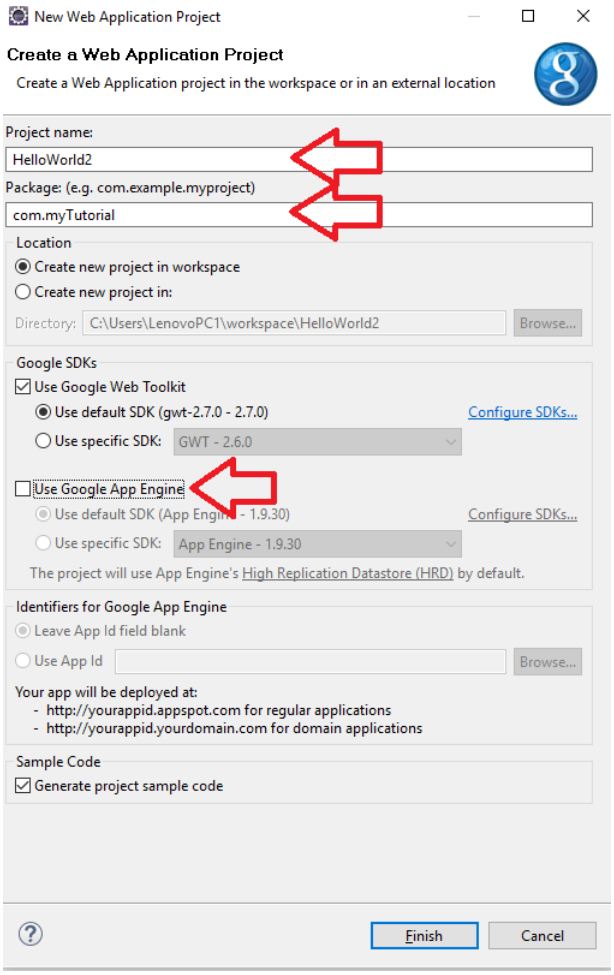
For at køre GWT, skal vi først have lavet os et program og hvad bedre end et "Hello World" til at starte med.

10.1 Hello World

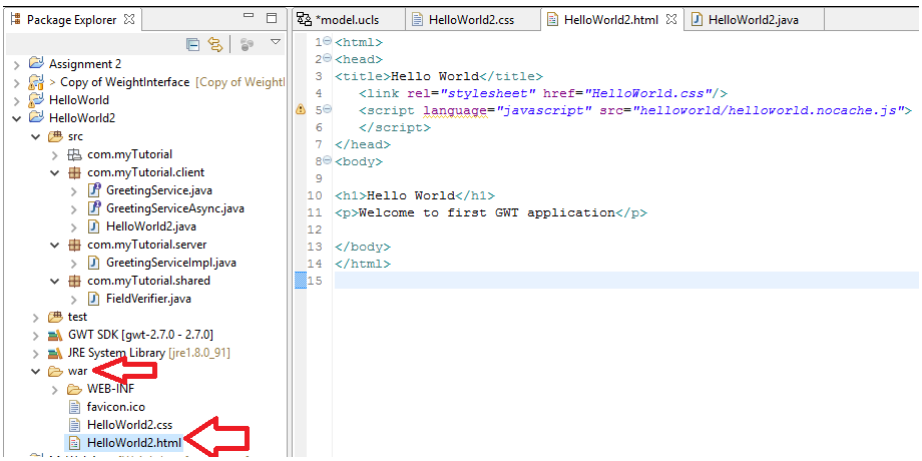
1) Klik på google iconet i top-menuen (repræsenteret med et lille hvidt g inde i en blå cirkel) og så på "New Web Application Project..."



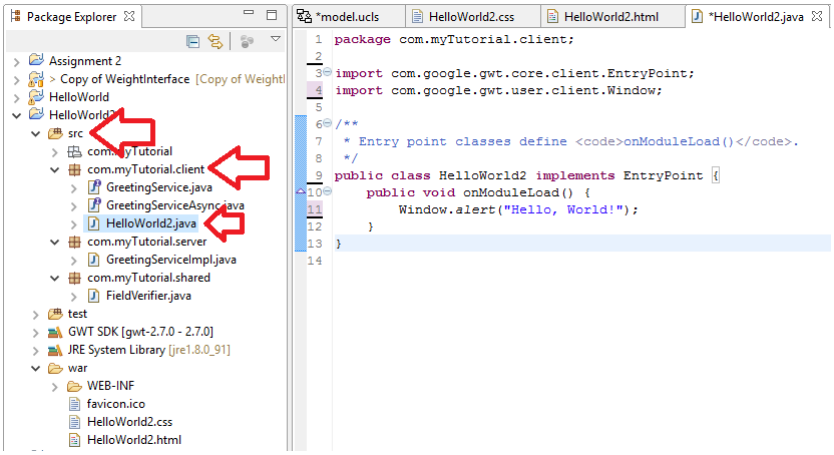
2) Navngiv projektet under ”Project name:”. Find på et package navn, det er ikke så vigtigt hvad du vælger, bare at det starter med com. efterfulgt at navnet eller flere navne hvis du har lyst (jeg bruger com.myTutorial). Derefter slå ”Use google App Engine” fra, da vi ikke har brug for den til det her projekt.



3) Du har nu et projekt ved det valgte navn. Projektet indeholder en SRC- og en WAR mappe. SRC er mappen med alle dine source filer i Java og WAR er mappen med alle dine web-applicationer. Gå ind under WAR folder, her finder du en HTML og en CSS fil. Åben HTML filen. Hvis du har lyst kan gå amok med HTML og CSS, men lad os holde det simpelt til at starte med (se figur for en simpel HTML).

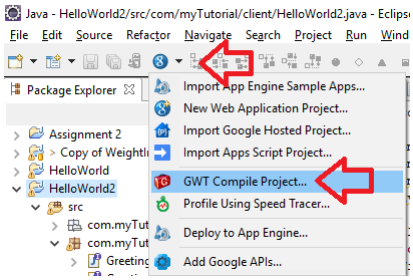


4) Gå nu ind under din SRC mappe, ind under client og åben java filen men samme navn som projektet. Erstat nu koden med det du ser på nedenstående billede.

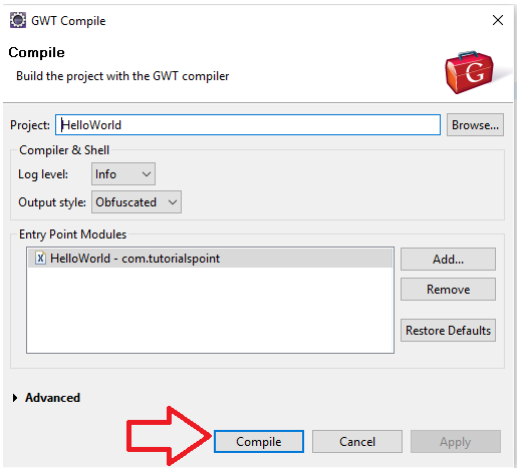


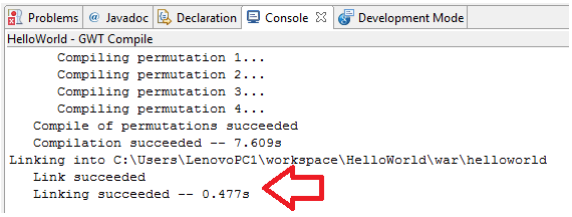
10.2 Hvordan man kører et GWT program

Nu har vi et projekt klar til at køre, så hvordan kører man et GWT projekt?
1) Vælg det GWT projekt du vil kører.
2) Klik endnu en gang på google iconet (blå cirkel med hvidt g i midten) og vælg "GWT Compile Project..."

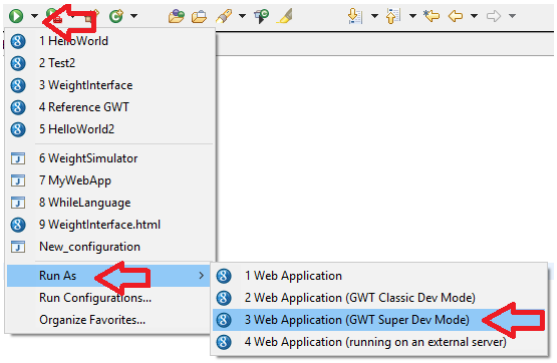


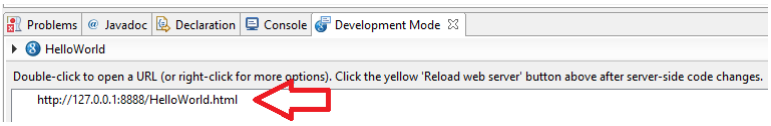
3) Et compile pop-up vindue vil komme frem. Klik på Compile i bunden og vent til der står "Link succeeded – 0.xxS" i konsolen





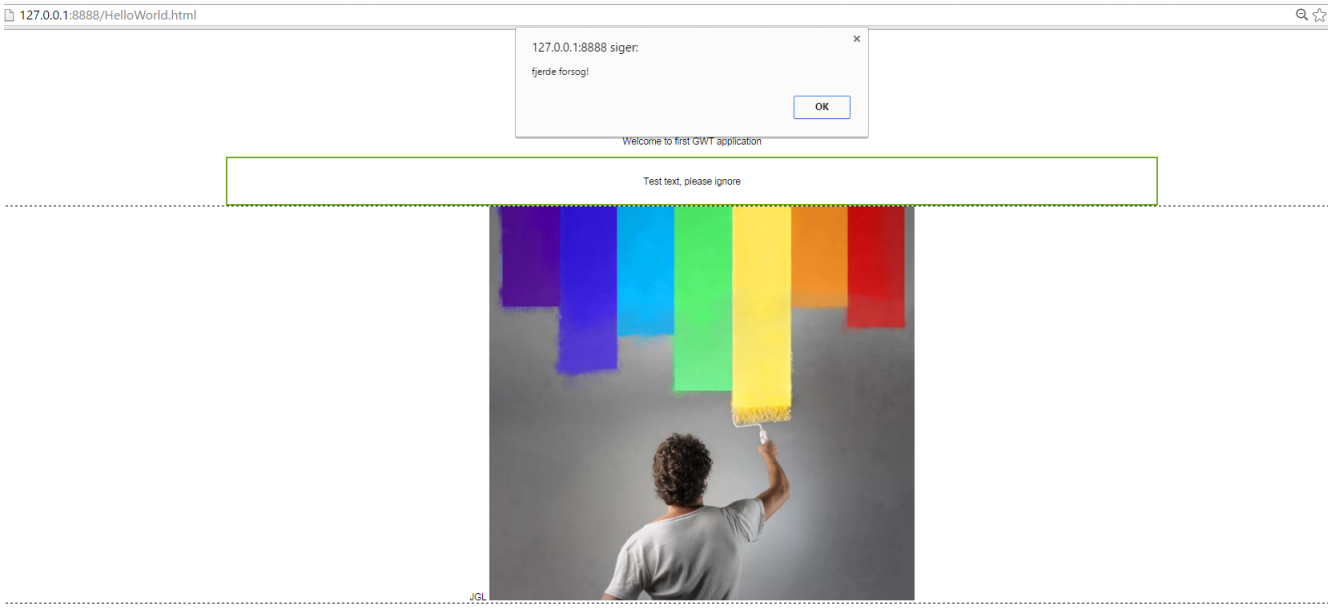
4) Gå nu op i toppen igen, og vælg drop-drop menuen til "afspilnings-iconet" (grøn cirkel med hvid trekant i midten). Vælg "Run As..." og vælg så "Web Application (GWT Super Dev Mode)" og vent til du får et link i konsolen. Hvis "Run As.." menuen er tom, prøv at klikke på dit GWT projekt over i exploreren først.





5) Højreklik linket og åben med dit valg af browser (Chrome er ikke default i Eclipse). Hvis dit valg af browser mangler, klik på "Add a browser" og tilføj den.
6) Du har nu åbnet dit projekt i en browser med HTML, CSS og Java (omskrevet til Javascript) kode.

Figure 31: Hvordan mit første Hello World endte med at se ud



11 Design

Hjemmeside design er vigtigt. Vi kender alle det at man kommer ind på en hjemmeside med et layout og farvekombination fra starten af 90'erne. Det er ikke ligefrem inviterende. Derfor var det vigtigt for os at have et ordentligt design, både hvad angår udsende og funktionalitet.

Vi besluttede derfor hurtigt at det skulle være så simpelt som muligt. Valget om en simpel hjemmeside blev taget da ingen af os havde arbejde med HTML, CSS eller Javascript før, hvilket gjorde at et simpelt design var mere inden for rimelighedens grænser at implementere.
Et simpelt design var også nemmere for folk, der normalt hellere ikke har brug for at være særlig kunsteriske, at designe noget der ser godt ud. Komplekse designs ville også tage mere tid at implementere og designe, som kunne ende med at straffe os hvis vejlederen sagde vi skulle starte forfra efter at have brugt lang tid på at designe noget stort og komplekst.

Efter at kigge efter inspirationer og farvekombinationer på nettet fik sat nogen grove byggesten for hvordan hjemmesiden skulle se ud. Vi besluttede for nogen store firkantede sektioner med skiftende farver, hvor hver sektion havde sin egen del af hjemmesiden (som vægt, batch, kontakt osv.). Vi tilføjede senere hen en navigations-bar, som altid var i toppen af hjemmesiden, for nem navigation rundt på hjemmesiden. For at gøre navigationen endnu nemmere tilføjede vi også nogen "Top" knapper som førte en op til toppen af siden, som man tit ser på hjemmesider.
Vi har igennem det meste af process taget højde for at man skulle kunne bruge hjemmesiden på alle devices, inkl. tablets og smartphones. Vi brugte derfor prøvet vores bedste altid at bruge em og procent enheder istedet for pixels.

Procent giver sig selv og er virkelig smart når man prøver at være crossplatform da den bare scaler iforhold til skærmstørrelse.
Em enheder afhænger af standard skriftstørrelsen man bruger, så stor skriftstørrelse gør en em enhed stor, hvilket for det meste passer godt ind i crossplatform.
Pixels fungerer ikke ordentligt da skærme har forskellige resolution, så pixel enheder kan se meget forskellige ud fra computer til smartphone.

Figure 32: Final layout på widescreen laptop

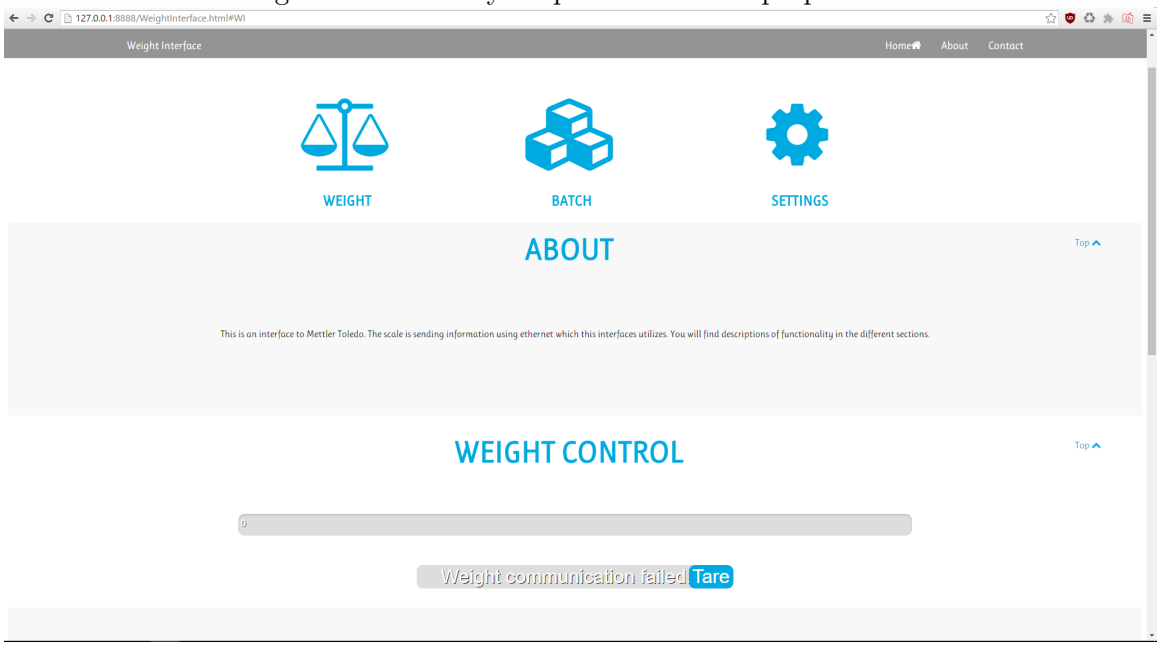
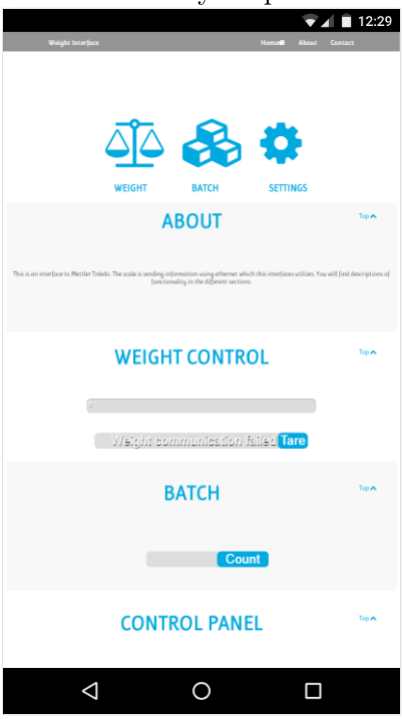


Figure 33: Final layout på en Nexus 5



12 Raspberry Pi

GWTs RPC kører rigtig godt på tomcat serverer, hvilket gjorde at vi igennem hele projekt tog brug af dem til at håndtere server siden af projektet. Det meste af tiden havde vi Tomcat serveren til at kører på Frederiks stationære derhjemme hver gang vi mødtes. Dette var dog ikke en langvarig løsning, da Frederik ikke kunne have sin computer kørende 24/7 og var også meget at dedikerer en hel stationær til bare en Tomcat server.

Vi fik derfor anskaffet os et par Raspberry Pi en den sidste uge og fik sat serveren over på et Linux system kørende på Pi'en, hvilket kan læses om i Frederiks rapport. Begrund af tidsmangel kunne vi ikke nå at finde et program der kunne køre vores WAR filer på en server som Tomcat. Det tætteste vi fandt var opsætningen af Windows IoT på en raspberry (som kan findes i biblografien), men der var ikke noget om en server kørende applikation ifølge vores søgninger.

13 Problemer Oplevet med GWT

Igennem projektet er der blevet oplevet både store og små problemer. Nogen var vigtige og tog tid at fikse, mens andre blev ignoreret da det højst var en mindre irritation.

13.1 Deling af filer - GIT

Kode delings programer er altid smarte at bruge når man koder, både når man arbejder alle eller i grupper. Dette projekt gjorde stor brug af det velkendte GIT for at fildele, men GWT og GIT er ikke de bedste venner når eclipse er mellem-manden.

Da projektet stadig var ungt, kodet vi mest af alt over skulder på en enkelt laptop, da det stadig var læringfasen hvad angår både vægten, Huawei-routeren og GWT. Da der for første gang blev gået fra en laptop til to, blev GIT taget i brug for at nemt at sende alt det det var blevet lavet indtil da. Men sådan lige at importere et GWT project i eclipse var svære end som så.

Hvis man henter fra GIT og bare sætter projektet ind i mappen, som man ville gøre med alle andre projekter, fejler eclipse. Et importeret projekt bliver ikke anset som et GWT projekt, så den kan ikke GWT compile.

Løsningen blev senere hen fundet til at være at lave et helt nyt GWT projekt i eclipse ved samme navn. Derefter tager man koden (ikke filerne) fra git og copy-paster over i dens korrensponderende GWT fil. Når dette er gjort for alle filerne kan man sætte GIT ligesom man plejer. Nu vil GIT altid sende de nyhentede filer over i GWT projektet, hvilket gør at den kan compile ordenligt.

13.2 Design

GWT er ikke ligefrem et venligt værktøj når det kommer til at implemterer sit personlige design-touch på hjemmesiden. De fleste stopklodser blev dog hurtigt løst med en hurtig google søgning, mens andre var mere besværlige at håndtere.

13.2.1 Knapper

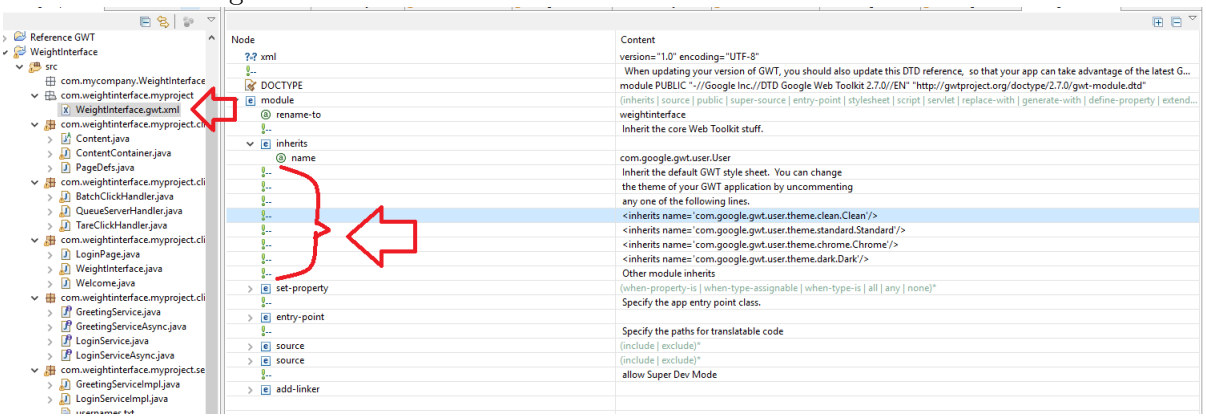
I de første mange itterationer af koden brugte vi GWTs egen styling når det kom til GWT widgets. Dette skyldes at da koden ikke var færdig, var det dumt at begynde at ændre på udsendet af knapperne, textfields osv. hvis de endte med at blive fjernet senere.

GWTs widgets default styling er meget kedeligt. windows 2000 styling knapper som kun er så store som teksten der står inden i knappen, så da resten af koden begyndte at blive finaliseret, måtte de ændres for at passe med resten af hjemmesidens layout og design.

Problemet opstod da der for første gang skulle afprøves en simpel CSS knap kode vi havde fundet på nettet, da vi ville teste det først. CSS styling var til stede på knappen, men som en "skal" uden om den originale knap. Så nu havde hjemmesidens knapper en kedelige grå firkanter i midten, med et farverigt og smooth layout uden om. Efter mange timers forsøg med forskellige CSS, blev der indset af det måske var GWT der forhindre sine knapper i at blive ændret.

Løsningen var en XML fil under Src/myproject/WeightInterface.gwt.xml hvor man skal udkommentere alle Inherit delene. Hvis man ikke udkommenterer dem, tvinger den GWT til at bruge sine standard design.

Figure 34: Det der skal udkommenteres i XML-filen



13.2.2 Jquery

Under et feedback møde med vejlederen, kom vi til den konklussion at vi havde for meget "vejledende" tekst under hvert modul på hjemmesiden. Under sectionerne *Weight, Batch og Controlpanel* havde vi lange vejledninger til hvordan man brugte den section af hjemmesiden, hvilket tog rigtigt meget plads, men også fokus væk fra de egentlige features som hjemmesiden tilbød. Det var stadig vigtigt at brugeren havde adgang til vejledende tekst i tilfældet af de ikke vidste hvordan det fungerede.

Figure 35: Eksmepel på jQuery kode til projektet

```
1  $(document).ready(function() {
2      $('.help').hover(function() {
3          $(this).show();
4      });
5  });
6
```

En af løsninger var at gøre det med jQuery, en slags undersprog til Javascript. Ovenstående figur viser den simple kode som ville løse problemet. Koden siger at hvis man mouseover et objekt med klassen '.help', så ville den vise teksten for den tilsvarende sektion. Et eksempel på hvordan det ville se ud kan ses på de næste to figurer.

Figure 36: Ingen mouseover [Help]

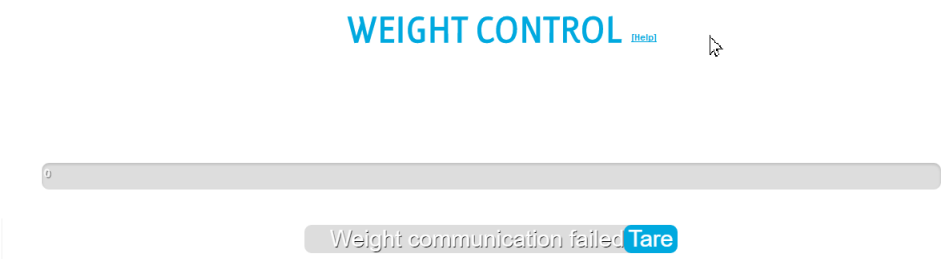
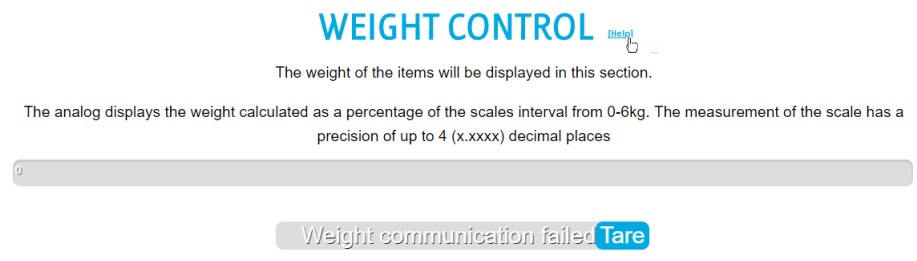


Figure 37: Mouseover [Help]



Vi fik det dog aldrig til at virke sammen med GWT. Højst sandsynligt er det GWTs Javascript der konflikter eller overskriver jQuery koden. Der er et ekstra plugin ved

navn gQuery som er jQuery i GWT, men begrund af tidsmangel måtte vi lade den feature ligge som den var.

13.3 Computer Frysninger

Dette problem skete kun på Windows laptopen. Nogen timer efter hver gang der var blevet arbejdet med GWT i Eclipse ville computeren semi-fryse. Med det menes at 75% af det man allerede havde åbent virkede, mens alt andet ved computeren stoppede med at fungere (inklusiv start menuen med Sluk/genstart funktionerne). Frysningerne skete først 2-6 timer efter man var færdig med at arbejde (så når man var kommet hjem igen). Problemet blev aldrig løst og endte altid i en hard reboot med sluk-tænd knappen på laptopen.

Dette betyder dog ikke at det er GWT der foresager frysningerne. Problemet kan ligge i Eclipse, den installeret version af Java eller bare laptopen selv.

14 Konklussion

We can hereby conclude that it is possible to make a connection between a website userinterface and a industri weight, which is on another connection with the help of GWT. GWTs ability to create fast connection without major work has been used and therefore verifies GWTs strengths if you just take time to learn how it works. Google has however shown lack of support and the numbers of users are dwindling, which means the enviorment is slowly dying.

Vi kan hermed se at det er muligt nemt at lave en forbindelse mellem en hjemmesides brugergrænseflade og en industrivægt, som er på et andet netværk, ved hjælp af GWT. GWTs evne til at oprette hurtige forbindelser uden det store arbejde er blevet brugt og har derfor vist GWTs styrker holder når man først har sat sig ind i hvordan det fungere. Google har dog vist mangel på fremtidig support og antal af bruger er faldende, så miljøet kan ses at gå i den forkerte retning.

15 Biblografi

References

[1] http://www.gwtproject.org/, Author: Google

[2] GWT - Create Application, Author: Tutorialspoint.com, Source: *http* : *//www.tutorialspoint.com/gwt/gwt_create_application.html*

[3] METTLER TOLEDO - Standard Interface Command Set, Author: Mettler Toledo *Pages: 1-98* Source: Stig

[4] METTLER TOLEDO Remote, Author: Mettler Toledo *Pages: 1-50* Source: Stig

[5] Set up a Raspberry Pi 2 or 3, Author: Microsoft Source: *https* : *//developer.microsoft.com/enus/windows/iot/win*10/*rpi*

[6] Codecademy kurser om HTML, CSS og Jquery, Author: Codecademy.com Team Source: *https* : *//www.codecademy.com/*

Ajax: A New Approach to Web Applications, Author: Jesse James Garrett Source: *https* : *//web.archive.org/web/*20080702075113/*http* : *//www.adaptivepath.com/ideas/essays/archives*

16 Appendix

17 Client

```
1  package com.weightinterface.myproject.client;
2
3  import com.google.gwt.user.client.ui.Composite;
4
5  /**
6   * Created by haanin on 5/2/16.
7   */
8  public abstract class Content extends Composite {
9      public abstract String getToken();
10     public abstract String getWindowTitle();
11 }
```

```
1  package com.weightinterface.myproject.client;
2
3  import com.google.gwt.event.logical.shared.
       ValueChangeEvent;
4  import com.google.gwt.event.logical.shared.
       ValueChangeHandler;
5  import com.google.gwt.http.client.Header;
6  import com.google.gwt.user.client.History;
7  import com.google.gwt.user.client.Window;
8  import com.google.gwt.user.client.ui.RootPanel;
9  import com.weightinterface.myproject.client.pages.
       LoginPage;
10 import com.weightinterface.myproject.client.pages.
       Welcome;
11 import com.weightinterface.myproject.client.pages.
       WeightInterface;
12
13 /**
14  * Created by haanin on 5/2/16.
15  */
16 public class ContentContainer implements
       ValueChangeHandler<String> {
17
18     @Override
19     public void onValueChange(ValueChangeEvent<
```

```
        String> event) {
20          String token = (String) event.getValue();
21          this.setContentByToken(token);
22      }
23
24      public void setContentByToken(String token) {
25          if (token.equals(PageDefs.WelcomeToken)) {
26              this.setContent(new Welcome());
27          } else if (token.equals(PageDefs.AppToken)
                ) {
28              this.setContent(new WeightInterface())
                    ;
29          } else if(token.equals(PageDefs.LoginToken
                )) {
30                  this.setContent(new LoginPage());
31          }
32      }
33
34      private void setContent(Content content) {
35          RootPanel contentRoot = RootPanel.get("
                content");
36          contentRoot.clear();
37          this.content = content;
38          History.newItem(content.getToken(), false)
                ;
39          // check for special initializations:
40          if (content.getToken().equals(PageDefs.
                AppToken)) {
41              ((WeightInterface) content).MyInit();
42          }
43          if (content.getToken().equals(PageDefs.
                LoginToken)) {
44              ((LoginPage) content).MyInit();
45          }
46          contentRoot.add(content);
47          Window.setTitle(content.getWindowTitle());
48          Window.scrollTo(0, 0);
49      }
50
51      private static ContentContainer myInstance =
            new ContentContainer();
52      public static synchronized ContentContainer
            getInstance() {
53          return myInstance;
54      }
```

```
55      private ContentContainer() {
56          History.addValueChangeHandler(this);
57      }
58
59      private Content content;
60
61  }
```

```
1   package com.weightinterface.myproject.client;
2
3   /**
4    * Created by haanin on 5/2/16.
5    */
6   public class PageDefs{
7       public static final String WelcomeToken = "
            WELCOME";
8       public static final String WelcomeTitle = "
            Welcome";
9       public static final String LoginToken = "LOGIN
            ";
10      public static final String LoginTitle = "Login
            ";
11      public static final String AppToken = "WI";
12      public static final String AppTitle = "
            WeightInterace";
13  }
```

17.1 Pages

17.1.1 Welcome

```
1   package com.weightinterface.myproject.client.pages
        ;
2
3   import com.google.gwt.core.client.EntryPoint;
4   import com.weightinterface.myproject.client.
        Content;
5   import com.weightinterface.myproject.client.
        PageDefs;
6
7   /**
8    * Created by haanin on 5/2/16.
9    */
```

```
10 public class Welcome extends Content implements
       EntryPoint {
11     @Override
12     public String getToken() {
13         return PageDefs.WelcomeToken;
14     }
15
16     @Override
17     public String getWindowTitle() {
18         return PageDefs.WelcomeTitle;
19     }
20
21     @Override
22     public void onModuleLoad() {
23         new LoginPage();
24         //new WeightInterface();
25     }
26 }
```

17.1.2 WeightSimulator

```
1
2  import java.net.ServerSocket;
3  import java.net.Socket;
4  import java.io.InputStream;
5  import java.io.OutputStream;
6  import java.io.IOException;
7  import java.io.PrintWriter;
8  import java.io.*;
9  import java.util.Random;
10
11
12 public class WeightSimulator implements Runnable
13 {
14     ServerSocket serverSocket = null;
15     Socket clientSocket = null;
16     Thread listener = null;
17
18
19     public WeightSimulator(int port) throws
           IOException
20     {
21         serverSocket = new ServerSocket(port);
22
```

```
23          listener = new Thread (this);
24
25          listener.start();
26      }
27
28
29      public void run()
30      {
31          boolean bError = false;
32          while(!bError)
33          {
34              try
35              {
36                  clientSocket = serverSocket.accept
                        ();
37                  synchronized (clientSocket)
38                  {
39                      try
40                      {
41                          PrintWriter out2 = new
                                PrintWriter(
                                getOutputStream(),true)
                                ;
42
43
44                          Random random = new Random
                                ();
45                          BufferedReader in =
46                                  new BufferedReader
                                        (
47                                          new
                                              InputStreamReader
                                              (
                                              clientSocket
                                              .
                                              getInputStream
                                              ()));
48                          String line;
49                          while((line = in.readLine
                                ()) != null)
50                          {
51                              System.out.println(
                                    line);
52                              if(line.contains("SI")
                                    )
```

```
53                                  {
54                                      out2.println("
                                           Weight: "+
                                           getRandomValue(
                                           random,0,6,4)+"
                                            kg");
55                                  }
56                                  else if(line.contains(
                                        "TI"))
57                                  {
58                                      out2.println("Tare
                                           ");
59                                  }
60                              }
61
62
63                          }
64                          catch (Exception e)
65                          {
66                              System.err.println("
                                   Exception in wait, "+ e
                                   .getMessage());
67                          }
68                          try
69                          {
70                              clientSocket.close();
71                          }
72                          catch (Exception e)
73                          {
74                              System.err.println("
                                   Exception in close, "+
                                   e.getMessage());
75                          }
76                      }
77                  }
78                  catch (IOException e)
79                  {
80                      bError = true;
81                  }
82              }
83
84              try
85              {
86                  serverSocket.close();
87              }
```

```
88          catch (Exception e)
89          {
90              System.err.println("Exception in close
                , "+ e.getMessage());
91          }
92      }
93
94      public static String getRandomValue(final
           Random random,
95      final int lowerBound,
96      final int upperBound,
97      final int decimalPlaces){
98
99      if(lowerBound < 0 || upperBound <= lowerBound
           || decimalPlaces < 0){
100         throw new IllegalArgumentException("Put
               error message here");
101     }
102
103     final double dbl =
104         ((random == null ? new Random() : random).
               nextDouble() //
105              * (upperBound - lowerBound))
106              + lowerBound;
107     return String.format("%." + decimalPlaces + "f
           ", dbl);
108
109 }
110
111
112     public void disconnect()
113     {
114         synchronized (clientSocket)
115         {
116             try
117             {
118                 clientSocket.notify();
119             }
120             catch (Exception e)
121             {
122                 System.err.println("Exception in
                       notify, "+ e.getMessage());
123             }
124         }
125     }
```

```
126
127
128     public void stop()
129     {
130         listener.interrupt();
131         try
132         {
133             serverSocket.close();
134         }
135         catch (Exception e)
136         {
137             System.err.println("Exception in close
                , "+ e.getMessage());
138         }
139     }
140
141
142     public InputStream getInputStream() throws
        IOException
143     {
144         if(clientSocket != null)
145         {
146             return(clientSocket.getInputStream());
147         }
148         else
149         {
150             return(null);
151         }
152     }
153
154
155
156     public OutputStream getOutputStream() throws
        IOException
157     {
158         if(clientSocket != null)
159         {
160             return(clientSocket.getOutputStream())
                ;
161         }
162         else
163         {
164             return(null);
165         }
166     }
```

```
167
168     public static void main(String[] args) throws
            IOException
169     {
170         WeightSimulator server = new
                WeightSimulator(23);
171     }
172 }
```

17.1.3 LoginPage

```
1  package com.weightinterface.myproject.client.pages
       ;
2
3  import com.google.gwt.core.client.GWT;
4  import com.google.gwt.event.dom.client.ClickEvent;
5  import com.google.gwt.event.dom.client.
       ClickHandler;
6  import com.google.gwt.event.dom.client.
       KeyDownEvent;
7  import com.google.gwt.event.dom.client.
       KeyDownHandler;
8  import com.google.gwt.user.client.Window;
9  import com.google.gwt.user.client.rpc.
       AsyncCallback;
10 import com.google.gwt.user.client.ui.*;
11 import com.weightinterface.myproject.client.
       Content;
12 import com.weightinterface.myproject.client.
       ContentContainer;
13 import com.weightinterface.myproject.client.
       PageDefs;
14 import com.weightinterface.myproject.client.
       services.LoginService;
15 import com.weightinterface.myproject.client.
       services.LoginServiceAsync;
16
17 import java.io.IOException;
18
19 /**
20  * Created by haanin on 5/2/16.
21  */
22 public class LoginPage extends Content {
23
```

```
24      private final LoginServiceAsync loginService =
            GWT.create(LoginService.class);
25      final Label loginLabel = new Label("WRITE YOUR
            ASSIGNED USERNAME");
26      final Button loginbutton = new Button("Login")
           ;
27      final TextBox username = new TextBox();
28      final VerticalPanel loginPanel = new
           VerticalPanel();
29      final VerticalPanel login2 = new VerticalPanel
           ();
30      final HTMLPanel contentPanel = new HTMLPanel("
           ");
31
32      public LoginPage() {
33      MyInit();
34      }
35
36      public void MyInit() {
37
38          loginbutton.setStyleName("login__submit");
39          RootPanel.get("content").setStyleName("
               background");
40          contentPanel.add(login2);
41          contentPanel.setStyleName("container");
42          loginPanel.setWidth("100%");
43          loginPanel.setHeight(Window.
               getClientHeight() + "px");
44          loginPanel.setHorizontalAlignment(
               HasAlignment.ALIGN_CENTER);
45          loginPanel.setVerticalAlignment(
               HasAlignment.ALIGN_MIDDLE);
46          loginPanel.add(contentPanel);
47          login2.setSpacing(70);
48          login2.setVerticalAlignment(HasAlignment.
               ALIGN_MIDDLE);
49          login2.setHorizontalAlignment(HasAlignment
               .ALIGN_CENTER);
50          login2.add(loginLabel);
51          login2.add(username);
52          login2.add(loginbutton);
53          RootPanel.get("content").add(loginPanel);
54          username.setStyleName("input");
55
56
```

```
57          loginbutton.addClickHandler(new
               ClickHandler() {
58              @Override
59              public void onClick(ClickEvent event)
                   {
60                  login();
61              }
62          });
63          username.addKeyDownHandler(new
               KeyDownHandler() {
64              @Override
65              public void onKeyDown(KeyDownEvent
                   event) {
66                  if(event.getNativeKeyCode()==13)
67                  {
68                      login();
69                  }
70              }
71          });
72      }
73
74      private void login() {
75          if(!username.getText().isEmpty())
76          {
77              try {
78                  loginService.checkLoggedIn(
                       username.getText().trim(), new
                       AsyncCallback<Boolean>() {
79                      @Override
80                      public void onFailure(
                           Throwable caught) {
81                          Window.alert(caught.
                               getLocalizedMessage());
82                      }
83
84                      @Override
85                      public void onSuccess(Boolean
                           usernameCorrect) {
86                          if(usernameCorrect)
87                          {
88                              ContentContainer.
                                   getInstance().
                                   setContentByToken(
                                   PageDefs.AppToken);
89                          }
```

```
90                              else
91                              {
92                                  Window.alert("Wrong
                                       username.");
93                                  username.setText("");
94                                  username.setFocus(true
                                       );
95                              }
96                          }
97                      });
98                  } catch (IOException e) {
99                      e.printStackTrace();
100                 }
101             }
102             else Window.alert("Write something please.
                    ");
103     }
104
105
106     @Override
107     public String getToken() {
108         return PageDefs.LoginToken;
109     }
110
111     @Override
112     public String getWindowTitle() {
113         return PageDefs.LoginTitle;
114     }
115
116
117 }
```

17.2 Services

17.2.1 GreetingService

```
1 package com.weightinterface.myproject.client.
    services;
2
3 import com.google.gwt.user.client.rpc.
    RemoteService;
4 import com.google.gwt.user.client.rpc.
    RemoteServiceRelativePath;
5
```

```
6  import java.io.IOException;
7
8  /**
9   * The client-side stub for the RPC service.
10  */
11 @RemoteServiceRelativePath("greet")
12 public interface GreetingService extends
       RemoteService {
13   //String greetServer(String name) throws
       IllegalArgumentException;
14   String sendCommandToWeight(String command)
       throws IOException;
15   String closeConnection() throws IOException;
16   String openConnection() throws IOException;
17 }
```

17.2.2 GreetingServiceAsync

```
1  package com.weightinterface.myproject.client.
       services;
2
3  import com.google.gwt.user.client.rpc.
       AsyncCallback;
4
5  import java.io.IOException;
6
7  /**
8   * The async counterpart of <code>GreetingService
        </code>.
9   */
10 public interface GreetingServiceAsync {
11
12   //String sendCommandToWeight(String s) throws
       IOException;
13   void sendCommandToWeight(String command,
       AsyncCallback<String> callback)
14        throws IOException;
15
16   void closeConnection(AsyncCallback<String>
       asyncCallback) throws IOException;
17   void openConnection(AsyncCallback<String>
       asyncCallback) throws IOException;
18
19 }
```

17.2.3 LoginService

```
1  package com.weightinterface.myproject.client.
       services;
2
3  import com.google.gwt.user.client.rpc.
       RemoteService;
4  import com.google.gwt.user.client.rpc.
       RemoteServiceRelativePath;
5
6  import java.io.IOException;
7
8  /**
9   * Created by haanin on 4/28/16.
10  */
11
12 @RemoteServiceRelativePath("login")
13 public interface LoginService extends
       RemoteService {
14     Boolean checkLoggedIn(String username) throws
           IOException;
15 }
```

17.2.4 LoginServiceAsync

```
1  package com.weightinterface.myproject.client.
       services;
2
3  import com.google.gwt.user.client.rpc.
       AsyncCallback;
4
5  import java.io.IOException;
6
7  /**
8   * Created by haanin on 4/28/16.
9   */
10 public interface LoginServiceAsync {
11     void checkLoggedIn(String username,
           AsyncCallback<Boolean> asyncCallback)
           throws IOException;
12 }
```

17.3 Handlers

17.3.1 BatchClickHandler

```
1  package com.weightinterface.myproject.client.
       handlers;
2
3  import com.google.gwt.event.dom.client.ClickEvent;
4  import com.google.gwt.event.dom.client.
       ClickHandler;
5  import com.google.gwt.user.client.Window;
6  import com.google.gwt.user.client.ui.Button;
7  import com.google.gwt.user.client.ui.TextBox;
8  import com.weightinterface.myproject.client.pages.
       WeightInterface;
9
10 /**
11  * Created by haaning on 3/31/16.
12  */
13 public class BatchClickHandler implements
       ClickHandler {
14
15     private Button countButton;
16     private TextBox count;
17     private TextBox currentWeightDisplay;
18     private boolean batchIsReady = false;
19     private double average;
20     private double weight;
21     private boolean isConnected = true;
22     private WeightInterface wi = null;
23
24     public BatchClickHandler(Button countButton,
           TextBox count, TextBox currentWeightDisplay
           , WeightInterface wi)
25     {
26         this.wi = wi;
27         this.countButton = countButton;
28         this.count = count;
29         this.currentWeightDisplay =
               currentWeightDisplay;
30     }
31
```

```
32
33      @Override
34      public void onClick(ClickEvent event) {
35          {
36
37              if(count.getText().matches("
                    [0-9]+\\.?\\,?[0-9]*") &&
                    isConnected())
38              {
39                  if(!batchIsReady)
40                  {
41                      average = wi.res / Double.
                            parseDouble(count.getText()
                            );
42                      countButton.setText("Done");
43                      count.setEnabled(false);
44                      Window.alert("1 weighs: "+
                            average+"\n"+"Place full
                            amount now and press the '
                            Done' button when you want
                            to quit batch counting.");
45
46                      batchIsReady = true;
47                  }
48                  else
49                  {
50                      count.setText("");
51                      countButton.setText("Count");
52                      count.setEnabled(true);
53                      batchIsReady = false;
54                  }
55
56              } else Window.alert("Cannot parse
                    input or is not connected.");
57          }
58      }
59
60      public double calculateAmount()
61      {
62          return wi.res / average;
63      }
64
65      public double getAverage()
66      {
67          return average;
```

```
68      }
69
70      public boolean isReady() {
71          return batchIsReady;
72      }
73
74      public double getCurrentWeight()
75      {
76          return Double.parseDouble(
               currentWeightDisplay.getText().
               substring(0,(currentWeightDisplay.
               getText().length()-3)));
77      }
78
79      public void setConnected(boolean status) {
           isConnected = status; }
80
81      public boolean isConnected() { return
           isConnected; }
82  }
```

17.3.2 QueueServerHandler

```
1   package com.weightinterface.myproject.client.
       handlers;
2
3   import com.google.gwt.dom.client.Style;
4   import com.google.gwt.event.dom.client.*;
5   import com.google.gwt.user.client.DOM;
6   import com.google.gwt.user.client.Timer;
7   import com.google.gwt.user.client.Window;
8   import com.google.gwt.user.client.rpc.
       AsyncCallback;
9   import com.weightinterface.myproject.client.pages.
       WeightInterface;
10
11  import java.io.IOException;
12
13  /**
14   * Created by haaning on 4/21/16.
15   */
16  public class QueueServerHandler implements
       ClickHandler, KeyUpHandler {
17
```

```
18
19      WeightInterface wi = null;
20      boolean isNegative = false;
21      private boolean alreadyClicked = false;
22
23      public QueueServerHandler(WeightInterface wi)
24      {
25      this.wi = wi;
26          wi.wb.setVisible(false);
27          doInit();
28      }
29
30      public void doInit() {
31          try {
32              weightPrompt();
33          } catch (IOException e) {
34              e.printStackTrace();
35          }
36      }
37
38      /**
39           * Fired when the user clicks on the
                 sendButton.
40           */
41          public void onClick(ClickEvent event) {
42              /*if(alreadyClicked)
43              {
44                  return;
45              }
46              else
47              {
48              try {
49                  alreadyClicked = true;
50                  weightPrompt();
51              } catch (IOException e) {
52                  e.printStackTrace();
53              }
54              }*/
55          }
56
57
58          public void timerOff()
59          {
60              wi.timerOn = false;
61              wi.wb.setText("ON");
```

```
62          }
63
64          public void onKeyUp(KeyUpEvent event) {
65              /*if (event.getNativeKeyCode() ==
                    KeyCodes.KEY_ENTER) {
66                  try {
67                      weightPrompt();
68                  } catch (IOException e) {
69                      e.printStackTrace();
70                  }
71              }*/
72          }
73
74          private void weightPrompt() throws
                IOException {
75              wi.getGreetingService().
                    sendCommandToWeight("SI", new
                    AsyncCallback<String>() {
76                  public void onFailure(Throwable
                        caught) {
77                      //DOM.getElementById("sk-
                            circle").getStyle().
                            setDisplay(Style.Display.
                            NONE);
78                      wi.currentWeightDisplay.
                            setVisible(true);
79                      wi.currentWeightDisplay.
                            setText("Weight
                            communication failed.");
80                      wi.wb.setText("ON");
81                      alreadyClicked = false;
82                      wi.currentWeightDisplay.
                            setVisibleLength(wi.
                            currentWeightDisplay.
                            getText().length()-4);
83                      try {
84                          weightPrompt();
85                      } catch (IOException e) {
86                          e.printStackTrace();
87                      }
88                  }
89
90                  public void onSuccess(String
                        weight) {
```

```
91                          //DOM.getElementById("sk-
                               circle").getStyle().
                               setDisplay(Style.Display.
                               NONE);
92                          isNegative = false;
93                          wi.wb.setText("OFF");
94                          char c = weight.charAt(7);
95                          if (!(c == ' ')) isNegative =
                               true;
96                          wi.res = Double.parseDouble(
                               weight.substring(8,14));
97                          if(wi.batch.isReady()){
98                              double batchdouble = wi.
                                   batch.calculateAmount()
                                   ;
99                              wi.count.setText(""+(int)
                                   batchdouble);};
100                         if(isNegative)
101                         {
102                             wi.currentWeightDisplay.
                                   setText("-"+wi.res+" kg
                                   ");
103                             DOM.getElementById("myBar"
                                   ).getStyle().setWidth
                                   (0, Style.Unit.PCT);
104                             DOM.getElementById("label"
                                   ).setInnerText("0%");
105                         } else
106                         {
107                             wi.currentWeightDisplay.
                                   setText(wi.res + " kg")
                                   ;
108                             DOM.getElementById("myBar"
                                   ).getStyle().setWidth(
                                   wi.res * 16.66, Style.
                                   Unit.PCT);
109                             DOM.getElementById("label"
                                   ).setInnerText(Double.
                                   toString(wi.res *
                                   16.66).substring(0, 6)
                                   + "%");
110                         }
111                         try {
112                             weightPrompt();
113                         } catch (IOException e) {
```

```
                            e.printStackTrace();
                        }
                    }
                });
        }
}
```

17.3.3 TareClickHandler

```
package com.weightinterface.myproject.client.handlers;

import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.rpc.AsyncCallback;
import com.weightinterface.myproject.client.pages.WeightInterface;

import java.io.IOException;

/**
 * Created by haaning on 4/14/16.
 */
public class TareClickHandler implements ClickHandler{

    WeightInterface wi = null;
    QueueServerHandler qh = null;

    public TareClickHandler(WeightInterface wi, QueueServerHandler qh)
    {
        this.wi = wi;
        this.qh = qh;


    }
    @Override
    public void onClick(ClickEvent event) {
        try {
            //qh.tareTimerOff();
```

```
30              wi.getGreetingService().
                   sendCommandToWeight("TI", new
                   AsyncCallback<String>() {
31                  @Override
32                  public void onFailure(Throwable
                       caught) {
33                      wi.windowAlert(caught.
                           getLocalizedMessage());  }
                           //qh.startTimer(); }
34
35                  @Override
36                  public void onSuccess(String
                       result) {
37                      if(result.equals("TI -"))
38                      {
39                          try {
40                              wi.getGreetingService
                                   ().
                                   sendCommandToWeight
                                   ("T", new
                                   AsyncCallback<
                                   String>() {
41                                  @Override
42                                  public void
                                       onFailure(
                                       Throwable
                                       caught) {
43
44                                  }
45
46                                  @Override
47                                  public void
                                       onSuccess(
                                       String result)
                                       {
48
49                                  }
50                              });
51                          } catch (IOException e) {
52                              e.printStackTrace();
53                          }
54                      }
55                      //qh.startTimer();
56                      /*wi.setWeightTextBar("0");
57                      wi.setAnalogBar(0);
```

```
58                      wi.setTimer(false);*/
59
60              }
61              });
62          }   catch (IOException e) {
63              e.printStackTrace();
64          }
65      }
66  }
```

18 Server

18.0.1 GreetingServiceImpl

```
1  package com.weightinterface.myproject.client.
       handlers;
2
3  import com.google.gwt.event.dom.client.ClickEvent;
4  import com.google.gwt.event.dom.client.
       ClickHandler;
5  import com.google.gwt.user.client.Window;
6  import com.google.gwt.user.client.ui.Button;
7  import com.google.gwt.user.client.ui.TextBox;
8  import com.weightinterface.myproject.client.pages.
       WeightInterface;
9
10 /**
11  * Created by haaning on 3/31/16.
12  */
13 public class BatchClickHandler implements
       ClickHandler {
14
15      private Button countButton;
16      private TextBox count;
17      private TextBox currentWeightDisplay;
18      private boolean batchIsReady = false;
19      private double average;
20      private double weight;
21      private boolean isConnected = true;
22      private WeightInterface wi = null;
23
24      public BatchClickHandler(Button countButton,
           TextBox count, TextBox currentWeightDisplay
           , WeightInterface wi)
```

```
25      {
26          this.wi = wi;
27          this.countButton = countButton;
28          this.count = count;
29          this.currentWeightDisplay =
               currentWeightDisplay;
30      }
31
32
33      @Override
34      public void onClick(ClickEvent event) {
35          {
36
37              if(count.getText().matches("
                   [0-9]+\\.?\\,?[0-9]*") &&
                   isConnected())
38              {
39                  if(!batchIsReady)
40                  {
41                      average = wi.res / Double.
                           parseDouble(count.getText()
                           );
42                      countButton.setText("Done");
43                      count.setEnabled(false);
44                      Window.alert("1 weighs: "+
                           average+"\n"+"Place full
                           amount now and press the '
                           Done' button when you want
                           to quit batch counting.");
45
46                      batchIsReady = true;
47                  }
48                  else
49                  {
50                      count.setText("");
51                      countButton.setText("Count");
52                      count.setEnabled(true);
53                      batchIsReady = false;
54                  }
55
56              } else Window.alert("Cannot parse
                   input or is not connected.");
57          }
58      }
59
```

```
60      public double calculateAmount()
61      {
62          return wi.res / average;
63      }
64
65      public double getAverage()
66      {
67          return average;
68      }
69
70      public boolean isReady() {
71          return batchIsReady;
72      }
73
74      public double getCurrentWeight()
75      {
76          return Double.parseDouble(
                currentWeightDisplay.getText().
                substring(0,(currentWeightDisplay.
                getText().length()-3)));
77      }
78
79      public void setConnected(boolean status) {
            isConnected = status; }
80
81      public boolean isConnected() { return
            isConnected; }
82  }
```

18.0.2 LoginServiceImpl

```
1   package com.weightinterface.myproject.client.
        handlers;
2
3   import com.google.gwt.event.dom.client.ClickEvent;
4   import com.google.gwt.event.dom.client.
        ClickHandler;
5   import com.google.gwt.user.client.Window;
6   import com.google.gwt.user.client.ui.Button;
7   import com.google.gwt.user.client.ui.TextBox;
8   import com.weightinterface.myproject.client.pages.
        WeightInterface;
9
10  /**
```

```
 * Created by haaning on 3/31/16.
 */
public class BatchClickHandler implements ClickHandler {

    private Button countButton;
    private TextBox count;
    private TextBox currentWeightDisplay;
    private boolean batchIsReady = false;
    private double average;
    private double weight;
    private boolean isConnected = true;
    private WeightInterface wi = null;

    public BatchClickHandler(Button countButton, TextBox count, TextBox currentWeightDisplay, WeightInterface wi)
    {
        this.wi = wi;
        this.countButton = countButton;
        this.count = count;
        this.currentWeightDisplay = currentWeightDisplay;
    }


    @Override
    public void onClick(ClickEvent event) {
        {

            if(count.getText().matches("[0-9]+\\.?\\,?[0-9]*") && isConnected())
            {
                if(!batchIsReady)
                {
                    average = wi.res / Double.parseDouble(count.getText());
                    countButton.setText("Done");
                    count.setEnabled(false);
                    Window.alert("1 weighs: "+average+"\n"+"Place full amount now and press the 'Done' button when you want
```

```
                        to quit batch counting.");
45
46                      batchIsReady = true;
47                  }
48                  else
49                  {
50                      count.setText("");
51                      countButton.setText("Count");
52                      count.setEnabled(true);
53                      batchIsReady = false;
54                  }
55
56              } else Window.alert("Cannot parse
                    input or is not connected.");
57          }
58      }
59
60      public double calculateAmount()
61      {
62          return wi.res / average;
63      }
64
65      public double getAverage()
66      {
67          return average;
68      }
69
70      public boolean isReady() {
71          return batchIsReady;
72      }
73
74      public double getCurrentWeight()
75      {
76          return Double.parseDouble(
                currentWeightDisplay.getText().
                substring(0,(currentWeightDisplay.
                getText().length()-3)));
77      }
78
79      public void setConnected(boolean status) {
            isConnected = status; }
80
81      public boolean isConnected() { return
            isConnected; }
82  }
```