JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll zur SEW-Übung

„Einheitenumwandlung mit JSF“

SEW
4AHITM 2015/16

Yunus Sari

Version 1.0

Note:

Betreuer: Michael Borko

Begonnen am 02. Mai 2016

Beendet am 05. Mai 2016

Inhaltsverzeichnis

1 Aufgabenstellung

Folge der Anleitung und erstelle einen Temperatur-Konverter und erweitere das Projekt, sodass auch Grad Kelvin unterstützt werden!

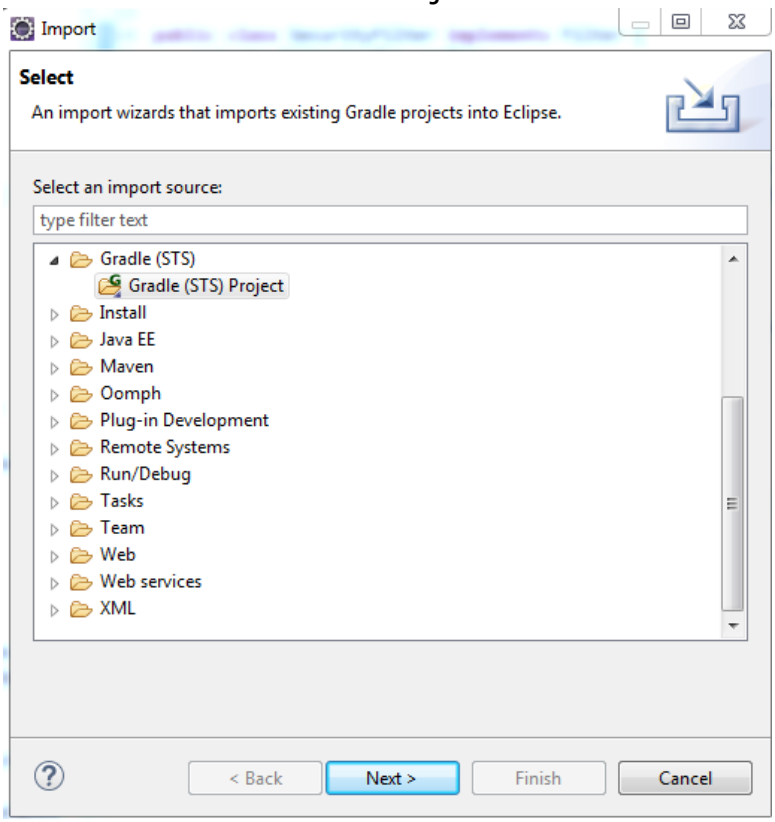
Abgabe: Protokoll (inkl. Kopf- und Fußzeile, Screenshots, Ergebnis, ...) + Code auf GitHub pushen

2 Aufwandschätzung

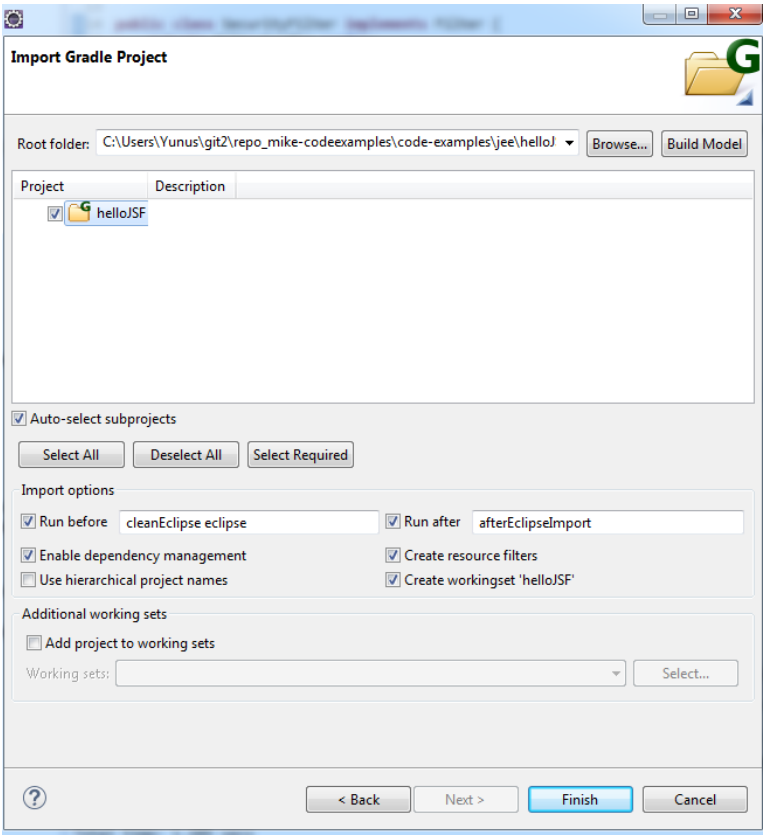
Arbeitspaket	**Geschätzter Aufwand**
Installation und Konfiguration der Umgebung	0 min – schon bereit
Erstellung der Grafischen Oberfläche	15 min
Erstellung der Funktionen	1 h 30 min
Protokoll	1 h 30 min
Summe	**3 h 15 min**

3 VORBEREITUNG

Importieren eines vorhanden Gradle Projekts:



Das jeweilige erstellte Projekt wird ausgewählt und mit finish importiert.



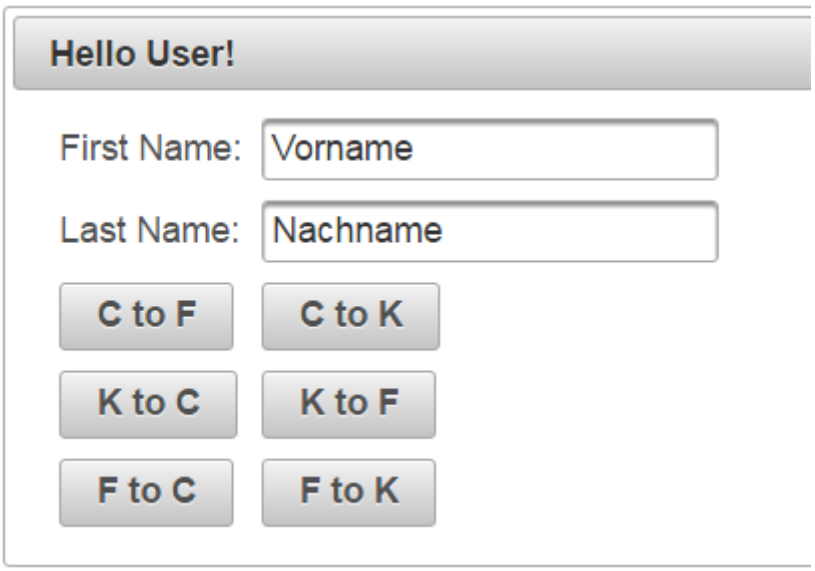
```
Jun 02, 2016 10:21:55 AM com.sun.faces.spi.InjectionProviderFactory createInstan
ce
INFORMATION: JSF1048: PostConstruct/PreDestroy-Annotationen vorhanden.  Verwalte
te Bean-Methoden, die mit diesen Annotationen markiert sind, lassen die entspre
henden Annotationen verarbeiten.
Jun 02, 2016 10:21:55 AM org.primefaces.webapp.PostConstructApplicationEventList
ener processEvent
INFORMATION: Running on PrimeFaces 5.2
10:21:55 INFO  Jetty 9.2.10.v20150310 started and listening on port 8080
10:21:55 INFO  Hello JSF runs at:
10:21:55 INFO    http://localhost:8080/
Press any key to stop the server.
Jun 02, 2016 10:42:16 AM com.sun.faces.config.ConfigureListener contextInitializ
ed
INFORMATION: Mojarra 2.2.8-15 ( 20160414-1926 unable to get svn info) f³r Kontex
t '' wird initialisiert.
Jun 02, 2016 10:42:17 AM com.sun.faces.spi.InjectionProviderFactory createInstan
ce
INFORMATION: JSF1048: PostConstruct/PreDestroy-Annotationen vorhanden.  Verwalte
te Bean-Methoden, die mit diesen Annotationen markiert sind, lassen die entspre
henden Annotationen verarbeiten.
Jun 02, 2016 10:42:17 AM org.primefaces.webapp.PostConstructApplicationEventList
ener processEvent
INFORMATION: Running on PrimeFaces 5.2
> Building 87% > :jettyRun
```

Mit „gradle jettyrun“ im Ordner wird es gestartet:

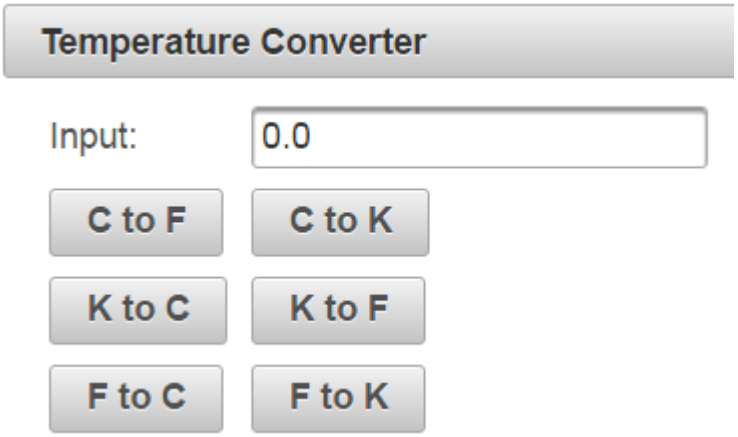
4 Umsetzung

Es wird das Beispiel von Hr. Prof. Borko genommen und nach unseren Wünschen und der Aufgabenstellung angepasst.
Das Beispiel HelloJSF wird modifiziert.

4.1 Umsetzung der Graphischen Oberfläche



Buttons wurden dupliziert und die richtigen Namen werden eingetragen.



Eingabe und Titel wurden angepasst.

4.2 Umsetzung der Funktionen

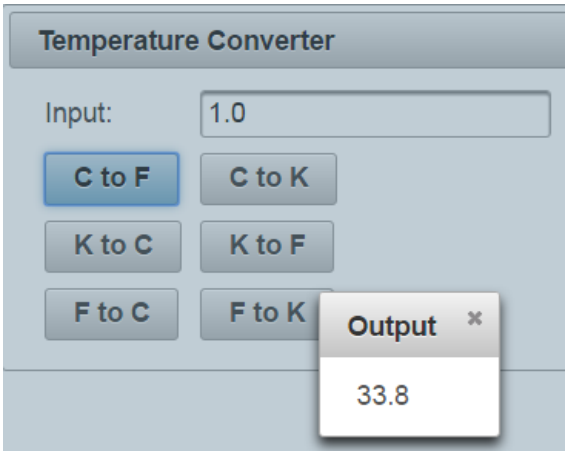
Für jede Konvertierung wurde eine Funktion in der HelloWorld.java Datei erstellt und in der hello.xhtml wurden zu jedem Button die Funktion hinzugefügt.

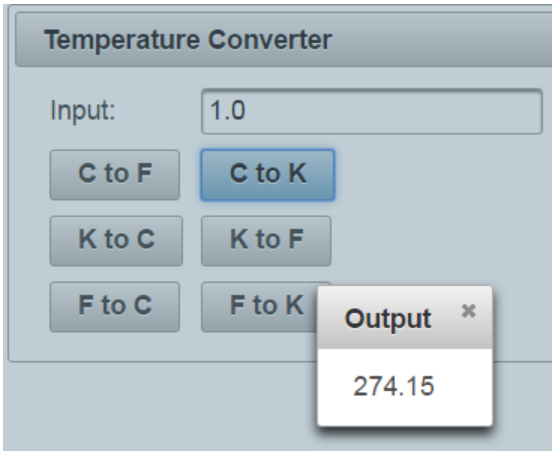
```
<p:dialog header="Output" widgetVar="dlg2" modal="true"
    resizable="false">
    <h:panelGrid id="display2" columns="1" cellpadding="4">
        <h:outputText value="#{helloWorld.ctof()}" />
    </h:panelGrid>
</p:dialog>

<p:dialog header="Output" widgetVar="dlg3" modal="true"
    resizable="false">
    <h:panelGrid id="display3" columns="1" cellpadding="4">
        <h:outputText value="#{helloWorld.ctok()}" />
    </h:panelGrid>
</p:dialog>

<p:dialog header="Output" widgetVar="dlg4" modal="true"
    resizable="false">
    <h:panelGrid id="display4" columns="1" cellpadding="4">
        <h:outputText value="#{helloWorld.ktoc()}" />
    </h:panelGrid>
</p:dialog>

<p:dialog header="Output" widgetVar="dlg5" modal="true"
    resizable="false">
    <h:panelGrid id="display5" columns="1" cellpadding="4">
        <h:outputText value="#{helloWorld.ktof()}" />
    </h:panelGrid>
</p:dialog>

<p:dialog header="Output" widgetVar="dlg6" modal="true"
    resizable="false">
    <h:panelGrid id="display6" columns="1" cellpadding="4">
        <h:outputText value="#{helloWorld.ftoc()}" />
    </h:panelGrid>
</p:dialog>

<p:dialog header="Output" widgetVar="dlg7" modal="true"
    resizable="false">
    <h:panelGrid id="display7" columns="1" cellpadding="4">
        <h:outputText value="#{helloWorld.ftok()}" />
    </h:panelGrid>
```

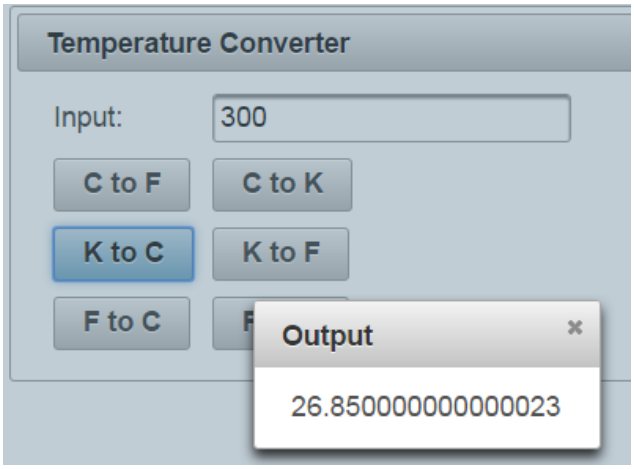
```
public String ctof() {
    double temp = Double.parseDouble(input)*9/5+32;
    output = String.valueOf(temp);
    return output;
}
public String ctok() {
    double temp = Double.parseDouble(input)+273.15;
    output = String.valueOf(temp);
    return output;
}
public String ktoc() {
    double temp = Double.parseDouble(input)-273.15;
    output = String.valueOf(temp);
    return output;
}
public String ktof() {
    double temp = Double.parseDouble(input)*9/5-459.67;
    output = String.valueOf(temp);
    return output;
}
public String ftoc() {
    double temp = (Double.parseDouble(input)-32)*5/9;
    output = String.valueOf(temp);
    return output;
}
public String ftok() {
    double temp = (Double.parseDouble(input)+459.67)*5/9;
    output = String.valueOf(temp);
    return output;
}
```

5 Demonstration

Hier sieht man einige Durchführungen der Buttonklicks.
Beim ersten Celsius nach Fahrenheit beim zweiten Celsius nach Kelvin und beim dritten Kelvin nach Celsius.







6 Versionisierung

Link:
https://github.com/ysari-tgm/jsf

7 Zeitaufzeichnung

Arbeitspaket	**Tatsächlicher Aufwand**
Installation und Konfiguration der Umgebung	0 min – schon bereit
Erstellung der Grafischen Oberfläche	45 min
Erstellung der Funktionen	1 h 15 min
Protokoll	1 h 0 min
Summe	**3 h 0 min**