



TEMPERATUR CONVERTER

SEW 2016

JAVA EE

Erstelle einen Temperatur-Konverter

David Sindle
4AHITM

Temperatur Converter

Inhalt

Vorab	1
Github Link	1
Vorbereitung	1
Gradle	1
Jetty	2
Programm	2
Programmierung	2
Converter.xhtml	2
Faces-config	3
Web.xml	3
Website	4

Vorab

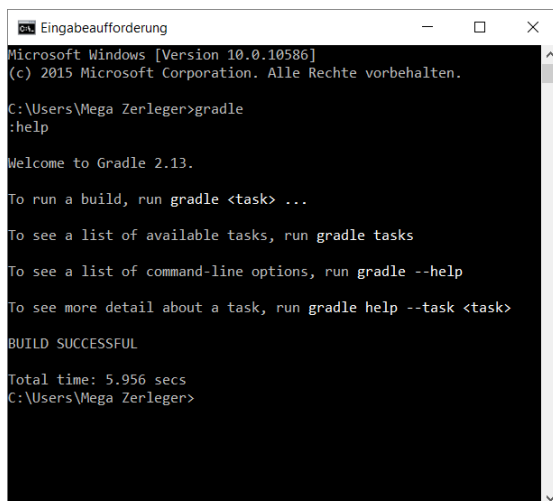
Github Link

<https://github.com/TgmSindl/TempConverter>

Vorbereitung

Gradle

Gradle starten



```
Eingabeaufforderung
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Mega Zerleger>gradle
:help

Welcome to Gradle 2.13.

To run a build, run gradle <task> ...

To see a list of available tasks, run gradle tasks

To see a list of command-line options, run gradle --help

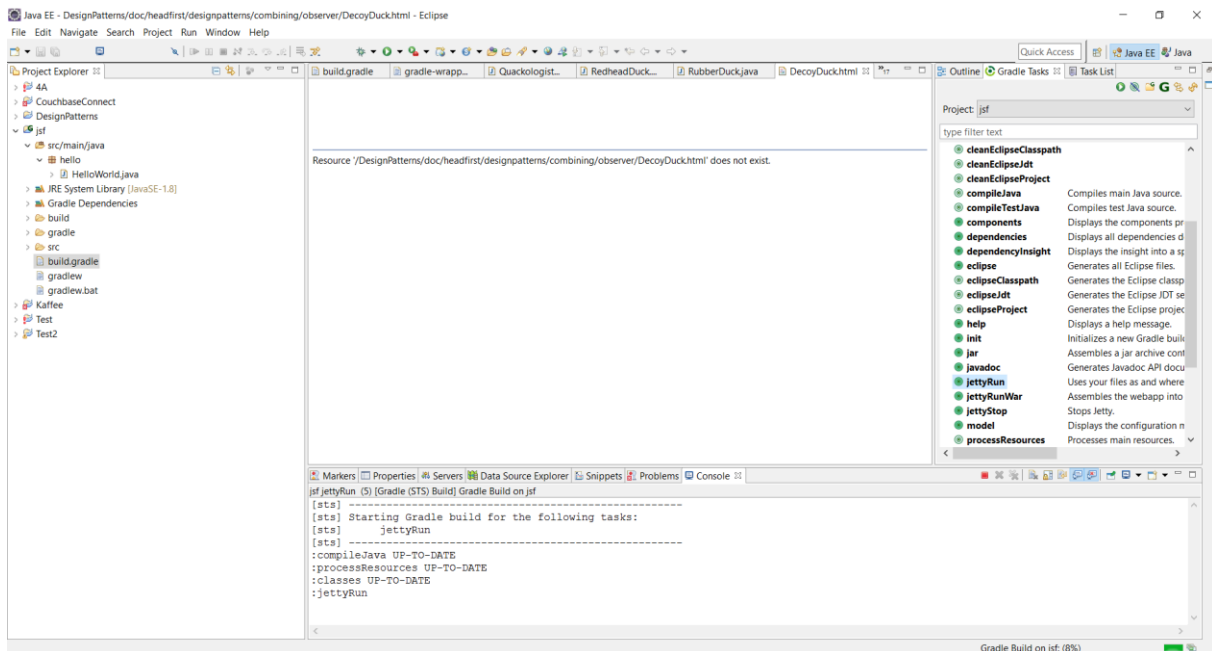
To see more detail about a task, run gradle help --task <task>

BUILD SUCCESSFUL

Total time: 5.956 secs
C:\Users\Mega Zerleger>
```

Jetty

Jetty in Eclipse mittels Jettyrun starten.



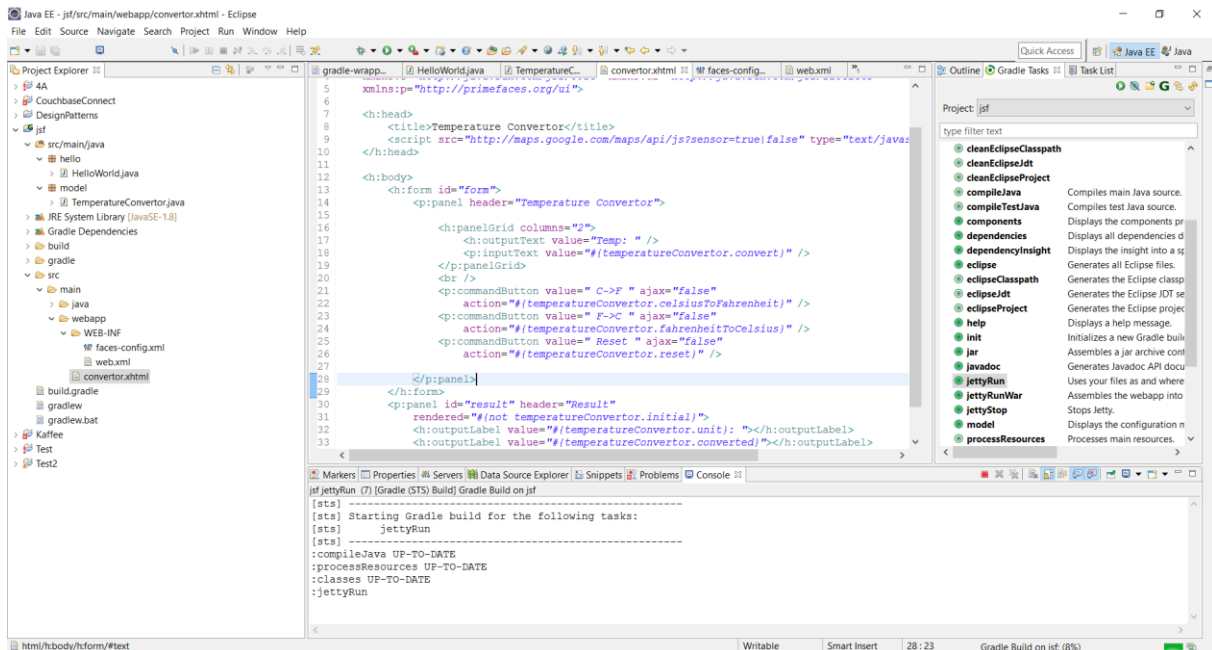
Programm

Ein betriebsbereites Programm starten, indem man das jsf file von den Code-Examples importiert.

Im Browser ansprechbar unter localhost:8080/tc

Programmierung

Converter.xhtml



Faces-config

The screenshot shows the Eclipse IDE with the `faces-config.xml` file open in the editor. The file is an XML configuration for a JSF application, defining a managed bean `TemperatureConverter` and a session bean `session`. The console output shows the Gradle build process for the `jettyRun` task, including the compilation of Java source files and the execution of the `jettyRun` task.

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
  version="2.2">
  <!--
  <managed-bean>
    <managed-bean-name>temperatureConverter</managed-bean-name>
    <managed-bean-class>model.TemperatureConverter</managed-bean-class>
    <managed-bean-scope>session</managed-bean-scope>
  </managed-bean>
  -->
</faces-config>
```

Console Output:

```
jettyRun (7) [Gradle (STS) Build] Gradle Build on jet
[sts] -----
[sts] Starting Gradle build for the following tasks:
[sts]     jettyRun
[sts] -----
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:jettyRun
```

Web.xml

The screenshot shows the Eclipse IDE with the `web.xml` file open in the editor. The file is an XML configuration for a web application, defining a `ConfigureListener` and a `TemperatureConverter` servlet. The console output shows the Gradle build process for the `jettyRun` task, including the compilation of Java source files and the execution of the `jettyRun` task.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">
  <listener>
    <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
  </listener>
  <!-- File(s) appended to a request for a URL that is not mapped to a
  web component -->
  <welcome-file-list>
    <welcome-file>converter.xhtml</welcome-file>
  </welcome-file-list>
  <!-- Define the JSF servlet (manages the request processing lifecycle
  forJavaServer) -->
  <servlet>
    <servlet-name>faces</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <!-- Map following files to the JSF servlet -->
  <servlet-mapping>
    <servlet-name>faces</servlet-name>
    <url-pattern>*.xhtml</url-pattern>
  </servlet-mapping>
</web-app>
```

Console Output:

```
jettyRun (7) [Gradle (STS) Build] Gradle Build on jet
[sts] -----
[sts] Starting Gradle build for the following tasks:
[sts]     jettyRun
[sts] -----
:compileJava UP-TO-DATE
:processResources UP-TO-DATE
:classes UP-TO-DATE
:jettyRun
```

Website

Nachdem alle Veränderungen wie in der PDF beschrieben durchgeführt wurden, kann man mittels localhost:8080/tc das ergebnis überprüfen, welches wie bei folgendem Screenshot aussieht.

Nun folgt die Erweiterung indem man bei converter.xhtml und bei der TemperaturConverter.java die richtigen Teile kopiert und angepasst werden

TemperaturConverter.java

```
public void fahrenheitToCelsius() {
    this.initial = false;
    this.unit="Celsius";
    this.converted = (convert - 32) / 1.8;
}
```

Converter.xhtml

```
<p:commandButton value=" C->F " ajax="false"
    action="#{temperatureConvertor.celsiusToFahrenheit}" />
```

Diese Zeilen kopieren und Anpassen und mit Hilfe der Seite:

<http://www.umrechnung.org/masseinheiten-temperatur-celsius-fahrenheit-kelvin/celsius-fahrenheit-umrechnung.htm> , habe ich herausgefunden, wie die Umrechnung erfolgt.