BMI Rechner

mit JavaEE

Inhalt

[1. Projekt 2](#_Toc31469203)

[2. Server 2](#_Toc31469204)

# Projekt

Dieses Projekt gehört zu meine Projekt Reihe, in der ich mit verschiedenen Technologien einen BMI Rechner realisieren möchte.

Bei diesem Projekt:

* Java
* Servlet
* JSP
* Tomcat
* Eclipse

# Projekt Start

Dieses Projekt soll in Eclipse realisiert werden.

## Struktur

Wir starten ein neues Dynamic Web Projekt in Eclipse.

Anschließen möchten wir einen MVC ansatz realisieren, deswegen erstellen wir folgende Package.

* Model
* Servlet – (Controll)
* Webcontent (durch Eclipse erstellt für View)

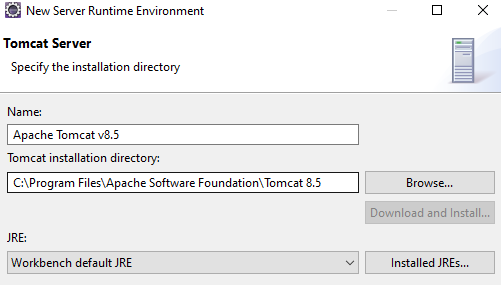
## Server

Um unser Projekt am Server zu bekommen müssen wir Eclipse informieren das es einen Webserver gibt.

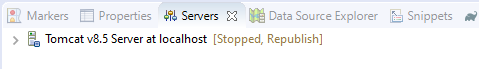
Dazu in Eclipse:

* Window
* Preferences
* Server
* Runtrime
* Environment Add
* Server auswählen
* Finisch

Wichtig ist Eclipse zu sagen wo er den Server findet, dieser muss natürlich davor installiert worden sein.

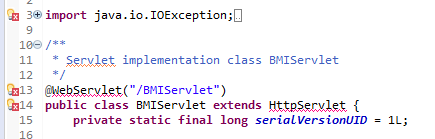


Zusätzlich fügen wir um unteren Bereich von Eclipse noch unseren Server hinzu, um diesen leichter starten zu können.



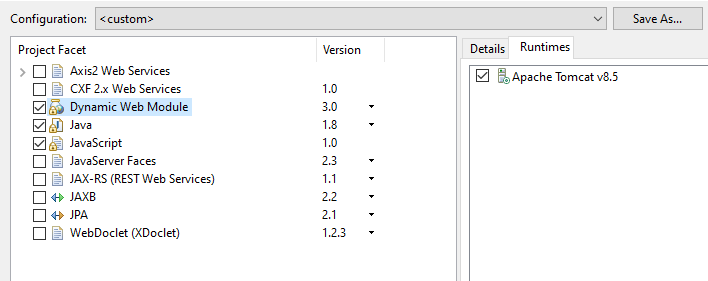
## Erstes Servlet

Danach lassen wir uns von Eclipse ein Servlet erstellen und nennen es BMIServlet.

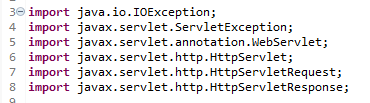


Wir erhalten gleich einmal Fehlermeldungen.

### Fehlerbehebung:



Ich habe vergessen den Server für das Projekt zu aktivieren.

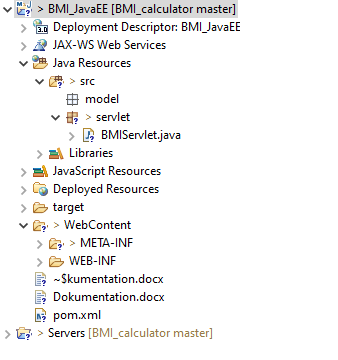


Zusätzlich habe ich das Servlet noch einmal neu erstellt, es müssen jetzt die javax Daten importiert werden.

### Maven:

Zusätzlich habe ich unser Projekt zu einen Maven Projekt gemacht, damit wir leichter unsere benötigten Daten erhalten.

Auch wird die Ordnerstruktur angepasst auf die bisher bekannte standart Ansicht, mit gradle wäre das natürlich auch möglich nur aufwendiger in Eclipse zu realisieren.



So jetzt können wir starten.

# Programmieren

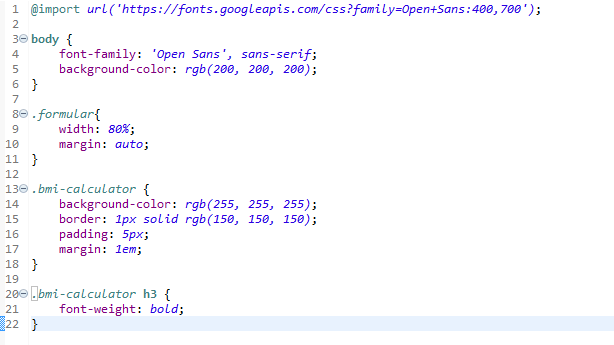
## View:



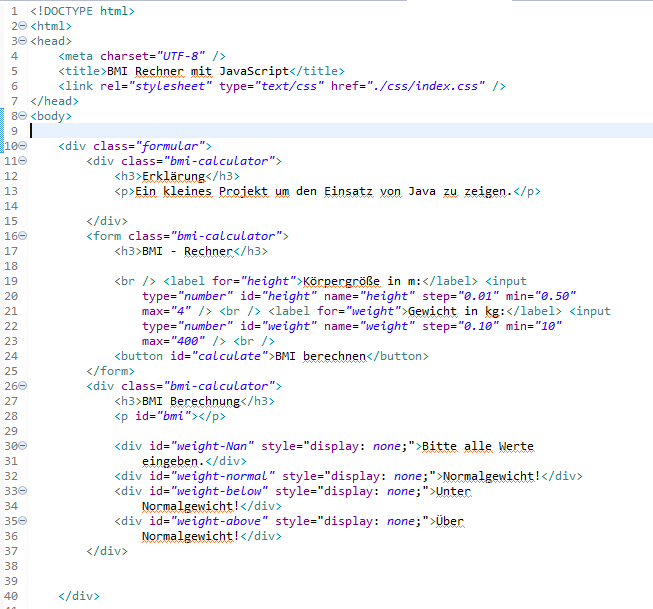
Wir erstellen jetzt 2 Ordner für css und js (falls benötigt) und eine index.html

Jetzt fügen wir den Code aus unseren vorigen Projekt mit ein und testen das Ganze.

### CSS:

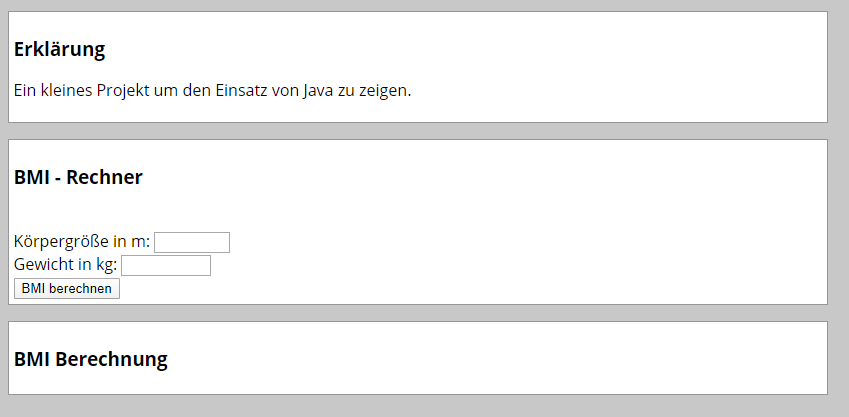


### HTML:





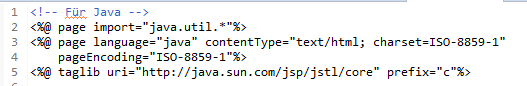
### Ergebnis



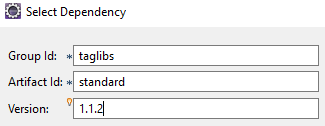
## Control Strukturen einbauen

### Abhängigkeiten

Hierfür mache ich aus der index.html eine index.jsp.



Damit das funktioniert muss ich natürlich die entsprechende Abhängigkeit laden.

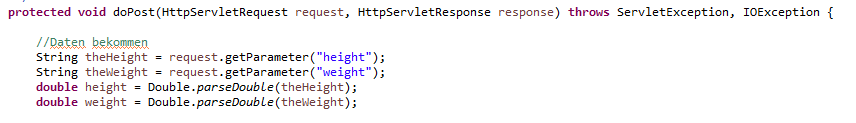


Desweitern wurde noch JUnit als Abhängigkeit hinzugefügt und dann Maven angewiesen das Projekt zu Updaten.

### Formular



Wir sagen den Formular jetzt unsere Anfrage geht an unser Servlet und wird per Post gesendet.



Danach können wir die Daten im Servlet abfragen und umwandeln.



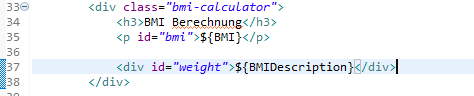
Jetzt habe ich unser Model aus dem JavaSE Projekt übernommen und führe die Berechnung damit aus.



Jetzt müssen wir noch die neuen Daten zur Verfügung stellen.



So hat die index.jsp jetzt die verfügbaren daten, mit der TagLib Bibliothek können wir darauf einfach zugreifen.



Mit diesen Tags erhält wir dann die Ergebnisse.

Zum Schluss sende ich den Benutzer zurück zur Index Seite.



Testen:

Beim Testen habe ich gesehen das ein Flüchtigkeitsfehler im System war.



Natürlich braucht das form die Anweisungen nicht das Div.



## Ergebnis:

