Maarten Lindt en Niels Borkes

onderzoeksverslag wicket

Titel : onderzoeksverslag wicket

Auteur : Maarten Lindt 616195 en Niels Borkes 616825

Course: DEA, leerjaar 2019/2020

Docent(en): Meron Brouwer

Plaats: Nijmegen, Doetinchem, Markelo

School: Hogeschool van Arnhem en Nijmegen

Datum: 5 mei 2020

|  |  |  |  |
| --- | --- | --- | --- |
| **WIJZIGINGSHISTORIE** | | | |
| *Datum* | *Versie* | *Beschrijving* | *Auteur* |
| 30 maart 2020 | 0.1 | Initiële versie | LDTM |
| 30 maart 2020 | 0.2 | Inleiding geschreven | LDTM |
| 3 april 2020 | 0.3 | Aangevuld | BKSN |
| 3 april 2020 | 1.0 | Final read and edit | LDTM |
| 5 mei 2020 | 1.1 | Verwerking feedback | BKSN & LDTM |

Contents

[1. Voorwoord 1](#_Toc36820782)

[2. Inleiding 2](#_Toc36820783)

[3. Wat is wicket? 3](#_Toc36820784)

[Het design pattern in Wicket 3](#_Toc36820785)

[Versie van het Framework 3](#_Toc36820786)

[4. Welke voordelen bied wicket in de Spotitube applicatie? 4](#_Toc36820787)

[Voordelen van wicket 4](#_Toc36820788)

[5. Hoe werken de componenten van wicket? 5](#_Toc36820789)

[5.1. Configuratie 5](#_Toc36820790)

[5.1.1. Implementatie 8](#_Toc36820791)

[5.2.1. Applicatie klasse van het type Java 8](#_Toc36820792)

[5.2.2. HomePage klasse van het type Java 9](#_Toc36820793)

[5.2.3. Een html pagina van het type html. 10](#_Toc36820794)

[5.2.4 CSS 10](#_Toc36820795)

[6. Toepassing op Spotitube 11](#_Toc36820796)

[6.1 HTML pagina 11](#_Toc36820797)

[6.2 CSS 12](#_Toc36820798)

[6.3. Java code 13](#_Toc36820799)

[6.3.1 webpage 13](#_Toc36820800)

[6.3.2 webpagina model 14](#_Toc36820801)

[6.3.3 De applicatie 15](#_Toc36820802)

[6.4 Wicket toepassing in Spotitube 16](#_Toc36820803)

[7. Conclusie 17](#_Toc36820804)

[8. Bronnen 18](#_Toc36820805)

# Voorwoord

Voor u ligt het wicket-onderzoek. Dit onderzoek was erg nuttig en we hebben er veel van geleerd. Maar we zaten ook vaak vast daarom willen we Meron Brouwer bedanken voor zijn hulp bij vragen. Verder willen we Meron ook bedanken voor de lessen van DEA. Zonder deze lessen en hulp was het mij waarschijnlijk niet gelukt. Tenslotte willen we ook Ruben Eppink bedanken voor het helpen als we weer eens vast zat. Lang leve puntkomma’s en hoofdletters.

We wensen u veel leesplezier met het lezen van dit verslag over wicket.

# Inleiding

Onze namen zijn Niels Borkes en Maarten Lindt, wij zijn beide studenten aan de HAN. Wij kregen de opdracht om onderzoek te doen naar hoe we Spotitube kunnen verrijken door iets toe te voegen of aan te passen.

Wij hebben ervoor gekozen om in dit verslag onderzoek te doen naar de vraag: Is de aangeleverde client applicatie van Spotitube te vervangen door een wicket applicatie?

Dit gaan we doen door de volgende deelvragen te beantwoorden:

* Wat is Wicket?
* Welke voordelen bied het gebruik van wicket in de Spotitube applicatie?
* Hoe werken de componenten van wicket?

De eerste 2 vragen zullen onderzoeken door online onderzoek. Voor de laatste deelvraag zullen we echter een klein prototype bouwen zodat we voor onszelf een gevoel kunnen krijgen hoe het werkt.

# Wat is wicket?

Apache Wicket is een web application framework voor Java dat zich richt op het versimpelen van het bouwen van web interfaces. Met Wicket is het mogelijk om een web applicatie te bouwen door alleen gebruik te maken van Java en XHTML, er hoeft dus geen Javascript of XML gebruikt te worden.

## Het design pattern in Wicket

Het framework werkt met het model-view-controller (MVC) pattern. In elke request cycle wordt de inkomende request gemapped naar een methode op een controller object, die hierop een response genereert, de controller haalt de data hiervoor uit een model om daarmee de view te kunnen aanpassen.

## Versie van het Framework

Het framework is origineel geschreven door Jonathan Locke in april 2004 en de eerste versie kwam in Juni 2005 uit. In de laatste release is er een overstap gemaakt van Java 8 naar Java 11 als minimaal benodigde versie van Java en ondersteund nu ook 12&13. Ook zijn ze over gegaan van jUnit 4 naar 5. Het gebruikmaken van de nieuwste release heeft voor ons voordeel want zo kunnen wij onze geleerde methodes toepassen op wicket.

# Welke voordelen bied wicket in de Spotitube applicatie?

In dit hoofdstuk zullen we kijken welke voordelen wicket bied.

## Voordelen van wicket[[1]](#footnote-1)

* De gehele configuratie is in java. Dus niet xml, json o.i.d.
* Informatie staat op de server. Dit is handig voor onderhoud, lagere download tijden.
* Geeft je gemakkelijk een AJAX object om de ui te updaten.
* Je hebt Models.

Models zijn een simpel contract voor het opvragen en het zetten van een waarde. De UI heeft dus Models om makkelijk data op te vragen en de Models zoeken uit hoe ze aan de data moeten komen. En waar ze eventueel data moeten opslaan bij bijvoorbeeld een invoerveld.

De voordelen van wicket Models is dat de Models pas aangeroepen worden als het echt nodig is. Dit voorkomt onnodig inladen van data.

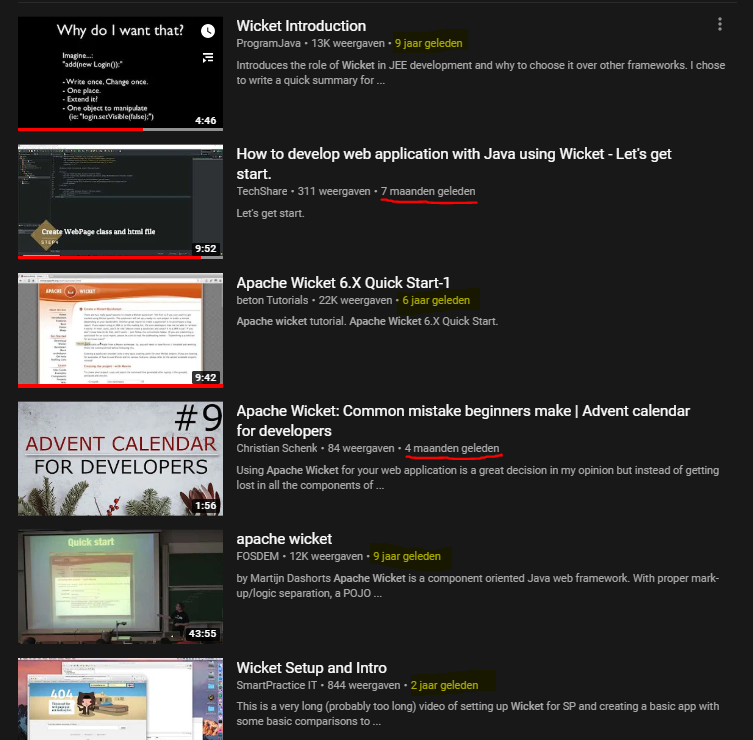
Het werkt namelijk als volgt: de Models worden gemaakt op de server. Voordat de client de HTML opvraagt. Vervolgens wordt de HTML gecreëerd volgens de Models en niet andersom.

* Maakt code reusable, je kunt een onderdeel één keer maken en overal implementeren.

# Hoe werken de componenten van wicket?

Om deze deelvraag te beantwoorden hebben we een prototype gebouwd. Vervolgens hebben we hieronder enkele belangrijke ondervindingen van wicket gezet. Eerst zullen we de configuratie uitleggen. Vervolgens de implementatie. Ook hebben we veel bronnen gebruikt. Zie hiervoor de bronnen in de bronnenlijst.

Deze informatie bijvoorbeeld door tutorials over het opzetten van wicket was meestal niet recent, zoals hieronder te zien zijn van de eerste 6 meest gerelateerde tutorials op YouTube er 4 ouder dan 2 jaar.



## Configuratie

Om een project te starten voer je de volgende command uit in je CMD, dit zal een project opstarten.

mvn archetype:generate -DarchetypeGroupId=org.apache.wicket -DarchetypeArtifactId=wicket-archetype-quickstart -DarchetypeVersion=9.0.0-M4 -DgroupId=nl.han.dea.nielsenmaarten -DartifactId=helloworld -DarchetypeRepository=https://repository.apache.org/ -DinteractiveMode=false

Hier moet je dan nog wel de groupid en artifact id aanpassen.

Dit archetype zet alles al voor je goed. Zo zet hij dependencies in je pom. Maakt hij een correcte web.xml aan. En geeft het je al een startcode om te kijken of alles werkt.

Maar als je het met de hand wilt doen moet je de volgende dependency toevoegen:

<dependency>  
 <groupId>org.apache.wicket</groupId>  
 <artifactId>wicket-core</artifactId>  
 <version>9.0.0-M4</version>  
</dependency>

We hebben hier voor versie 9.0.0-M4 gekozen omdat deze junit 5 en java 12 en 13 ondersteund.

En ook moet je enkele aanpassingen doen in je <build> settings.

Je moet namelijk de volgende regels toevoegen:

<resources>  
 <resource>  
 <filtering>false</filtering>  
 <directory>src/main/resources</directory>  
 </resource>  
 <resource>  
 <filtering>false</filtering>  
 <directory>src/main/java</directory>  
 <includes>  
 <include>\*\*</include>  
 </includes>  
 <excludes>  
 <exclude>\*\*/\*.java</exclude>  
 </excludes>  
 </resource>  
</resources>

Vervolgens moet je dan ook een web.xml aanmaken. Hier een voorbeeld van de gehele web.xml van helloworld.

<?xml version="1.0" encoding="ISO-8859-1"?>  
<web-app 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://www.oracle.com/webfolder/technetwork/jsc/xml/ns/javaee/web-app\_3\_1.xsd"  
 version="3.1">  
  
 <display-name>helloworld</display-name>  
  
  
 <filter>  
 <filter-name>wicket.helloworld</filter-name>  
 <filter-class>org.apache.wicket.protocol.http.WicketFilter</filter-class>  
 <init-param>  
 <param-name>applicationClassName</param-name>  
 <param-value>nl.han.dea.nielsenmaarten.WicketApplication</param-value>  
 </init-param>  
 </filter>  
  
 <filter-mapping>  
 <filter-name>wicket.helloworld</filter-name>  
 <url-pattern>/\*</url-pattern>  
 </filter-mapping>  
</web-app>

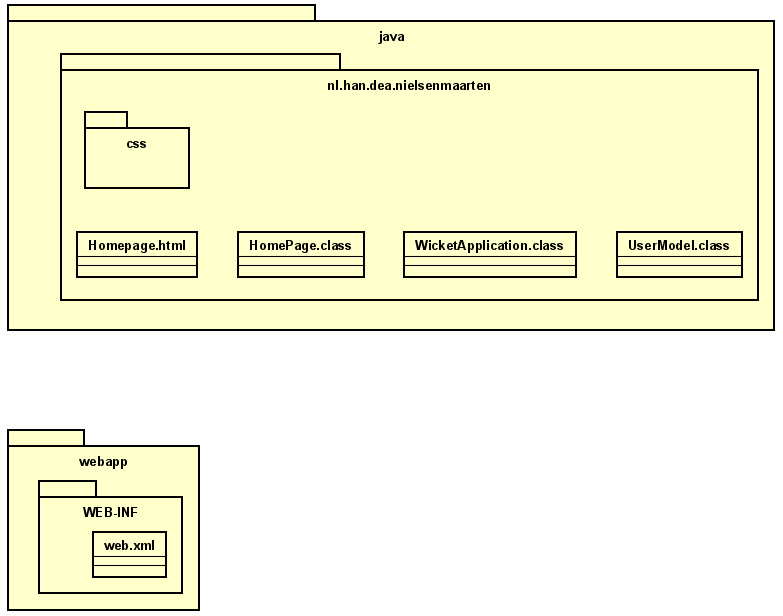
Hier hebben wij de filter en filtermapping aan toegevoegd. Een filter is een object dat verantwoordelijk is van het filteren van een aanvraag naar een resource en/of voor het filteren van een response van een resource.   
Filters bevatten de volgende attributen die je toe moet voegen:

* Filter name.   
  Hier zet je de naam van de filter neer. Let op deze moet hetzelfde zijn als de filter naam in de filter mapping.
* Filter class  
  Hier zeg je wat voor filter het is. In dit geval is het een wicketFilter.
* Param name  
  Dit geeft aan dat de value die je hierna toekent een applicatieklasse is.
* Param value  
  Waar je je applicatie hebt staan.

Filter mapping bevat het volgende:

* Filter name   
  De naam van de filter die hij moet gaan mappen.(heb je hierboven dus aangemaakt).
* URL pattern  
  Naar welke URL jee het moet sturen. In dit geval gewoon naar de root.

## 5.2 Implementatie

Voordat de individuele classes worden uitgelegd laten we de structuur van de applicatie zien in een diagram. 

Figuur 1 Structuur applicatie

### 5.2.1. Applicatie klasse van het type Java

Als je eenmaal de configuratie goed hebt moet je een Java klasse maken die de applicatie wordt. Deze klasse zal er in grote lijnen het zelfde uitzien voor ieder project.

package nl.han.dea.nielsenmaarten;  
  
import org.apache.wicket.markup.html.WebPage;  
import org.apache.wicket.protocol.http.WebApplication;  
  
public class WicketApplication extends WebApplication  
{  
@Override  
 public Class<? extends WebPage> getHomePage()  
 {  
 return HomePage.class;  
 }  
  
}

Dit voorbeeld is het minimale wat je nodig hebt.

In dit voorbeeld vallen een paar dingen op namelijk:

* Je override de functie getHomePage. De naam zegt het al hier geef je je homepage terug.
* getHomePage is van het type (Class<? Extends WebPage>) is. Dit geeft aan dat het een klasse terug geeft die WebPage extend. Je hoeft dus niet specifiek te zijn welke klasse dit is.

WebPage is een klasse die je import van wicket. Hierdoor ziet wicket dat het een web pagina is.

### 5.2.2. HomePage klasse van het type Java

Vervolgens moet je een homepageklasse aanmaken. Hieronder zie je een minimaal voorbeeld:

package nl.han.dea.nielsenmaarten;  
  
import org.apache.wicket.markup.html.basic.Label;  
import org.apache.wicket.markup.html.WebPage;  
  
public class HomePage extends WebPage {  
  
  
 public HomePage(){  
  
 add(new Label("hello", "alloo wereld"));  
  
 // *TODO Add your page's components here* }  
}

Je ziet dat hier een klasse aan wordt gemaakt die WebPage extend.

Ook zie je dat in de constructor de functie add wordt aangeroepen. Deze functie is van wicket en kent een waarde toe aan iets in je html file.

In dit geval zeg je bij het label hallo moet de message alloo wereld zijn. Hierover wordt meer verteld in het hoofdstuk: HomePage klasse van het type Java

### 5.2.3. Een html pagina van het type html.

Tot slot heb je zoals eerder al gezegd een html pagina nodig. Dit fungeert als bouwtekening voor waar je iets in wilt voeren vanuit de java klasses. Hieronder een voorbeeld van een website met alleen een beetje tekst:

<!DOCTYPE html>  
<html xmlns:wicket="http://wicket.apache.org">  
 <head>  
 <meta charset="utf-8" />  
 <title>Apache Wicket Quickstart</title>  
  
 </head>  
 <body>  
 <p>  
 <span wicket:id="hello">message</span>  
 </p>  
  
  
 </body>  
</html>

Je ziet dat het een redelijk standaard html opmaak heeft. Dit wil zeggen de tags zijn volgens een standaard html pagina. Echter zijn er wat dingen anders, namelijk:

* In de html tag zie je dat het volgende is toegevoegd: xmlns:wicket=<http://wicket.apache.org> dit zorgt ervoor dat de editor geen probleem maakt van eventuele onbekende tags zoals wicket:id etc.
* Het volgende wat je kun zien is dat in de span een wicketid wordt aangemaakt. Hierdoor kan wicket in de webpagina klasse een bedoelde waarde neerzetten in de span.

### 5.2.4 CSS

Je kunt ook Css toevoegen aan het Wicket project, dit kan op verschillende manieren. Wij hebbben het zo gedaan:

<wicket:link>  
 <link rel="stylesheet" type="text/css" href="css/style.css"/>  
 <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css"/>  
 <link href='https://fonts.googleapis.com/css?family=Amaranth' rel='stylesheet'/>  
 <link href='https://fonts.googleapis.com/css?family=Open Sans' rel='stylesheet'/>  
 <link href='https://fonts.googleapis.com/css?family=Raleway' rel='stylesheet'/>  
</wicket:link>

Binnen de header moet je de wicket:link zetten, als je dit niet doet dan weet Wicket niet dat het zijn responsebility is om af te handelen en denkt dat je het door de container wil laten doen en zal hier dus ook niet naar kijken.

# 6. Toepassing op Spotitube

## 6.1 HTML pagina

Voor de pagina tot toepassing op spotitube te laten komen moest er een login-form gemaakt worden met hierin een knop en twee velden: gebruikersnaam en wachtwoord. Dit login-form is doormiddel van CSS van de benodigde styling voorzien. Zo ziet de code er nu uit:

<!DOCTYPE html>  
<html xmlns:wicket="http://wicket.apache.org">  
 <head>  
 <meta charset="utf-8" />  
 <title>Spotitube</title>  
  
</head>  
 <wicket:link>  
 <link rel="stylesheet" type="text/css" href="css/style.css"/>  
 <link rel="stylesheet" type="text/css" href="css/bootstrap.min.css"/>  
 <link href='https://fonts.googleapis.com/css?family=Amaranth' rel='stylesheet'/>  
 <link href='https://fonts.googleapis.com/css?family=Open Sans' rel='stylesheet'/>  
 <link href='https://fonts.googleapis.com/css?family=Raleway' rel='stylesheet'/>  
 </wicket:link>  
<body>  
<div class = "textCenter">  
 <h2>  
 <span wicket:id="loggedin">message</span>  
 </h2>  
</div>  
<div class="container login-container">  
 <div class="row">  
 <div class="col-md-6 login-form-1">  
 <h3>Inloggen</h3>  
 <form wicket:id="form">  
  
 <div class="form-group">  
 <input type="text" class="form-control textIn" placeholder="Uw email of gebruikersnaam\*" wicket:id="username"/>  
 </div>  
 <div class="form-group">  
 <input type="password" class="form-control" placeholder="Uw wachtwoord\*" wicket:id="password" />  
 </div>  
 <div class="form-group textCenter">  
 <input type="submit" class="btnSubmit textCenter" wicket:id="submit" />  
 </div>  
 </form>  
 </div>  
 <div class="col-md-6 login-form-2">  
  
 <h3>Spotitube</h3>  
 <form>  
 <div class="form-group">  
 <div class="stepByStep textCenter">  
 <h5>Luister hier al uw muziek!<br></h5>  
 <p>1. Log nu in<br>  
 2. Maak playlists<br>  
 3. Vul ze met uw faforiete muziek<br>  
 4. Luister ernaar met veel plezier</p>  
 </div>  
 </div>  
 </form>  
 </div>  
 </div>  
</div>  
</body>  
</html>

## 6.2 CSS

Zo als je kan zien hebben we aan de html pagina 3 fonts toe gevoegd en twee stylesheets de invoervelden zijn required en hiervan is het veld voor de gebruikersnaam een text-field en voor het wachtwoord een password-field.

Dit is het resultaat:

A screenshot of a cell phone

Description automatically generated

Zonder de css bestanden zal het loginscherm er zo uit zien:

A screenshot of a cell phone

Description automatically generated

## 6.3. Java code

Zoals al gespecificeerd in het hoofdstuk hoe werkt wicket moet je ook java code hebben. Namelijk een model, een webpage en een main applicatie.

### 6.3.1 webpage

Op de achtergrond worden de elementen van de html pagina aan Java geknoopt met deze pagina code:

public class HomePage extends WebPage {  
 public HomePage() {  
  
 UserModel userModel = new UserModel();  
  
 Form<?> form = new Form("form");  
   
 TextField<String> username = new TextField<String>("username", new PropertyModel<String>(userModel, "username"));  
  
 PasswordTextField password = new PasswordTextField("password", new PropertyModel<String>(userModel, "password"));  
  
 Label loggedin = new Label("loggedin", "u bent ingelogd.");  
 loggedin.setVisible(false);  
   
 Button button = new Button("submit") {  
 @Override  
 public void onSubmit() {  
 super.onSubmit();  
 userModel.login();  
 if(userModel.getToken() != null){  
 loggedin.setVisible(true);  
 System.*out*.println(userModel.getToken());  
 }  
  
 }  
 };  
 add(loggedin);  
 add(form);  
  
 form.add(username);  
 form.add(password);  
 form.add(button);  
 }  
}

In dit stuk van HomePage.java wordt als eerste een Usermodel aangemaakt deze class wordt gebruikt als een plaats om de invoer op te slaan. Als tweede worden voor alle elementen uit de html pagina objecten gemaakt, zoals: Label, Form, TextField, PasswordTextfield en Button. Op het laatst worden alle aangemaakte objecten toegevoegd.

In de twee TextFields wordt het element met dezelfde naam als het wicket:id in de html pagina aan een van de van de variabelen van UserModel gekoppeld. Met deze variabelen kan vervolgens gewerkt worden door ze in dit geval met gegevens uit de database te vergelijken.

Je ziet ook loggedin.setVisible, dit zorgt ervoor dat component van de html niet wordt laten zien totdat je het om true zet.6.3.2 webpagina model

Zoals je al gezien hebt, wordt in de webpagina klasse verschillende keren een model aangeroepen. Dit model ziet er als volgt uit:

package nl.han.dea.nielsenmaarten;  
  
  
import com.github.openjson.JSONObject;  
  
import java.io.\*;  
import java.net.HttpURLConnection;  
import java.net.MalformedURLException;  
import java.net.ProtocolException;  
import java.net.URL;  
  
public class UserModel implements Serializable {  
  
 private String username;  
 private String password;  
 private String token;  
  
 public String getUsername() {  
 return username;  
 }  
  
 public void setUsername(String username) {  
 this.username = username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public void login() {  
 String jsonInputString = "{\"user\": \"" + username + "\", \"password\": \"" + password + "\" }";  
 try {  
 URL url = new URL("http://localhost:8081/login");  
 HttpURLConnection con = (HttpURLConnection) url.openConnection();  
 con.setRequestMethod("POST");  
 con.setRequestProperty("Content-Type", "application/json;");  
 con.setRequestProperty("Accept", "application/json");  
 con.setDoOutput(true);  
 try (OutputStream os = con.getOutputStream()) {  
 byte[] input = jsonInputString.getBytes("utf-8");  
 os.write(input, 0, input.length);  
 }  
 BufferedReader in = new BufferedReader(  
 new InputStreamReader(con.getInputStream()));  
 String inputLine;  
 StringBuffer response = new StringBuffer();  
 while ((inputLine = in.readLine()) != null) {  
 response.append(inputLine);  
 }  
 in.close();  
 JSONObject myResponse = new JSONObject(response.toString());  
 this.token = myResponse.getString("token");  
  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
  
 }  
  
 public String getToken() {  
 return token;  
 }  
}

hierboven zie je verschillende dingen. Je ziet dat er een user, een wachtwoord en een token worden toegekend aan dit model. Hier kan de webpagina bij.

Vervolgens zijn hier getters en setter voor gemaakt. Om ervoor te zorgen dat webpage de namen van de variabelen niet hoeft te kennen.

Maar dan zie je ineens een heel blok code in login. Deze zullen we dan ook stapsgewijs uitleggen.

1. Je begint met een Json inputstring. Deze string bevat eigenlijk al een Json met wat je wilt gaan versturen naar Spotitube.

2. Vervolgens zie je dat de URL word gepakt om later naar te callen.

3. Dan wordt er een connectie gestart. Hier worden headers en het andere benodigde aan toegevoegd.

4. Vervolgens wordt de post geprobeerd met de juiste UTF.

5. Dan lezen we de output.

6. Tot slot wordt door de volgende twee regels:

JSONObject myResponse = new JSONObject(response.toString());  
 this.token = myResponse.getString("token");

Eerst het response omgezet naar een JSON object en dan wordt het token opgeslagen onder de variabele token.

### 6.3.3 De applicatie

De applicatie is verder hetzelfde gebleven als beschreven in het hoofdstuk hoe werkt wicket. Dit is dus als volgt:

public class WicketApplication extends WebApplication  
{  
 @Override  
 public Class<? extends Page> getHomePage() {  
 return HomePage.class;  
 }  
}

## 6.4 Wicket toepassing in Spotitube

Zoals je kon zien in de bovenstaande koppen is Wicket een framework om een web applicatie te maken. Wij gaan een wicket applicatie maken met een httpserver om zo tegen Spotitube te kunnen praten, dit gaat de huidige Spotitube client vervangen om in te loggen.   
Dit heeft als voordeel dat we nu niet meer afhankelijk zijn van de client die school levert en dus de coupling verminderd door niet meer gebruik te hoeven maken van een 3rd party software.

Ook hebben wij gekozen voor een webapplicatie die tegen Spotitube praat met requests, omdat dit makkelijker te implementeren was dan een wicket applicatie verweven in de Java-ee container. Dit zou wel kunnen maar zou meer tijd kosten aan verder onderzoek en experimenteren.

# 7. Conclusie

Kortom het is goed mogelijk een web interface te maken met behulp van wicket. Toen we eenmaal de informatie hadden konden we direct aan de slag met een testapplicatie om de componenten te leren kennen(voornamelijk het mvc model en de specifieke wicket functies). Vervolgens konden we dan ook vlot wicket aan Spotitube knopen. Het gehele proces ging makkelijker dan gedacht. Dit komt waarschijnlijk door de eenvoudige en eenduidige werking van wicket. Ook hebben we enkele voordelen van wicket gevonden. Het grootste voordeel was misschien wel het mvc model.

# 8. Bronnen

AIM\_ HAN. (z.d.). Onderzoeksopdracht Enterprise Applications. Geraadpleegd op 40 maart 2020, van https://onderwijsonline.han.nl/elearning/lesson/QN41kYLy

Del Bene, A., Grigorov, M., Hufe, C., kroemer, C., Bartl, D., Bors, P., … Rhode, J. (z.d.). 3 Why should I learn Wicket? 6.x. Geraadpleegd op 30 maart 2020, van https://ci.apache.org/projects/wicket/guide/6.x/guide/whyLearn.html

ProgramJava. (2010). Wicket Introduction [Videobestand]. Geraadpleegd van https://www.youtube.com/watch?v=Vt1v1P6deA4

Wikipedia contributors. (2019, 12 september). Apache Wicket. Geraadpleegd op 30 maart 2020, van https://en.wikipedia.org/wiki/Apache\_Wicket

beton Tutorials. (2014, 3 mei). Apache Wicket 6.X Form components and Bindings-2. Geraadpleegd op 30 maart 2020, van <https://www.youtube.com/watch?v=Q9rUaQ02nVw&t=449s>

Working with Wicket models. (2011, 14 november). Geraadpleegd op 30 maart 2020, van https://cwiki.apache.org/confluence/display/WICKET/Working+with+Wicket+models#WorkingwithWicketmodels-WhatareWicketModels?

1. ProgramJava. (2010). [↑](#footnote-ref-1)