Windpark REST

Timmelmayer Bernhard, 05.11.2020

Aufgabenstellung

Entwickeln Sie einen Simulator der die Daten der Windkraftanlage generiert. Es ist dabei zu achten, dass die Daten realistisch sind und im Zusammenhang mit einer entsprechenden Einheit erzeugt werden. Die Daten der Windkraftanalage sollen ueber einer REST Schnittstelle veroeffentlicht werden. Die Schnittstelle verwendet standardmaessig das JSON Format und kann optional auf XML umgestellt werden. Die Schnittstelle soll mit einer einfachen Applikation getestet werden. Dabei sollen die Daten mit Hilfe von HTML und JQuery "konsumiert" und in einer Tabelle dargestellt werden.

Umsetzung (GKÜ)

Man musste sich die ZIp-Datei von elearning entpacken. Man muss die pom.xml-File austauschen. Danach öffnet man das Projekt in IntelliJ, die Applikation wird initialisiert. Um nun die Daten anzeigen zu lassen musste man folgenden Code schreiben:

```
private WindengineService service;

@RequestMapping("/")
public String windengineMain() {
    String mainPage = "This is the windengine application! (DEZSYS_GK72_WINDPARK) <br/><br/>" +
            "<a href='http://localhost:8080/windengine/001/xml'>Link to windengine/001/xml</a><br/>" +
            "<a href='http://localhost:8080/windengine/001/json'>Link to windengine/001/json</a><br/>" +
            "<a href='consumer.html'>See data</a><br/>";
    return mainPage;
}

@RequestMapping(value="/windengine/{windengineID}/xml", produces = {"application/xml"})
public WindengineData windengineDataXML( @PathVariable String windengineID ) {
    return service.getWindengineData( windengineID );
}

@RequestMapping(value="/windengine/{windengineID}/json", produces = {"application/json"})
public WindengineData windengineDataJSON( @PathVariable String windengineID ) {
    return service.getWindengineData(windengineID);
}
```

In der Annotation @RequestMapping (value, produces), muss man die Art des Outputs angeben, in unserem Fall xml oder json. Der Output wird dann beim angegebenen Pfad (value="") angezeigt.
Nachdem man diese Schritte gemacht hat, muss man in die Konsole `mvn spring-boot:run` eingeben und die Seite aufrufen (http://localhost:8080/).

Je nachdem auf welchen Link man klickt bekommt man die Daten in Form von XML oder JSON:

```
▼<WindengineData>
   <windengineID>001</windengineID>
   <timestamp>2020-11-06 11:55:30.322</timestamp>
   <windspeed>17.34</windspeed>
   <unitWindspeed>kmH</unitWindspeed>
   <temperature>17.82</temperature>
   <unitTemperature>C</unitTemperature>
   <power>1705.59</power>
   <unitPower>kwH</unitPower>
   <blindpower>6.27</blindpower>
   <unitBlindpower>kwH</unitBlindpower>
   <rotationspeed>184.07</rotationspeed>
   <unitRotationspeed>uM</unitRotationspeed>
   <bladeposition>4.0</bladeposition>
   <unitBladeposition>grad</unitBladeposition>
 </WindengineData>
```


```
{"windengineID":"001","timestamp":"2020-11-06
11:55:45.096","windspeed":28.1,"unitWindspeed":"kmH","temperature":-39.37,"unitTemperature":"C","po
wer":235.43,"unitPower":"kwH","blindpower":29.81,"unitBlindpower":"kwH","rotationspeed":25.54,"unit
RotationSpeed":"uM","bladeposition":42.0,"unitBladeposition":"grad"}
```


Umsetzung (GKV)

Unter /resources/templates findet man eine consumer.html Datei. In dem Projekteordner erstellt man einen Ordner namens public, in diesen Ordner kopiert man die consumer.html Datei. Die Datei muss in dem public-Ordner sein, weil sie sonst nicht aufgerufen werden kann.

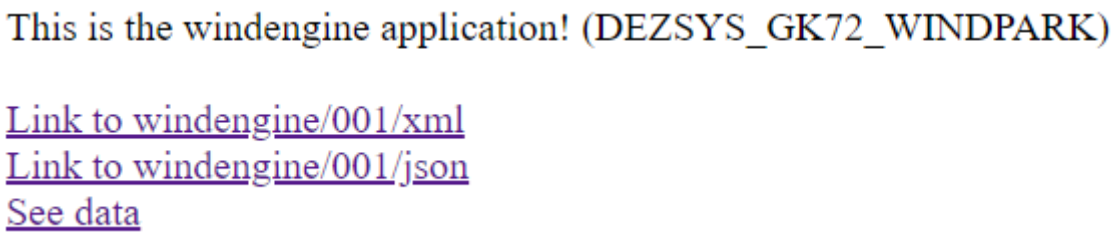
```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Getting Started: Serving Web Content</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>
<script>
    $(document).ready(function() {
        $.ajax({
            url: "http://localhost:8080/windengine/001/json"
        }).then(function(data) {
            $('#windengineID').append(data.windengineID);
            $('#timestamp').append(data.timestamp);
            $('#windspeed').append(data.windspeed);
            $('#unitWindspeed').append(data.unitWindspeed);
            $('#temperature').append(data.temperature);
            $('#unitTemperature').append(data.unitTemperature);
            $('#power').append(data.power);
            $('#unitPower').append(data.unitPower);
            $('#blindpower').append(data.blindpower);
            $('#unitBlindpower').append(data.unitBlindpower);
            $('#rotationspeed').append(data.rotationspeed);
            $('#unitRotationspeed').append(data.unitRotationspeed);
            $('#bladeposition').append(data.bladeposition);
```

```
            $('#unitBladeposition').append(data.unitBladeposition);
        });
    });
</script>
<body>
<table>
    <tr>
        <th>ID</th>
        <th>Time</th>
        <th>Windspeed</th>
        <th>UnitWindspeed</th>
        <th>Temperature</th>
        <th>UnitTemperature</th>
        <th>Power</th>
        <th>UnitPower</th>
        <th>Blindpower</th>
        <th>UnitBlindpower</th>
        <th>Rotationspeed</th>
        <th>UnitRotationspeed</th>
        <th>Bladeposition</th>
        <th>UnitBladeposition</th>
    </tr>
    <tr>
        <td><span id="windengineID"></span></td>
        <td><span id="timestamp"></span></td>
        <td><span id="windspeed"></span></td>
        <td><span id="unitWindspeed"></span></td>
        <td><span id="temperature"></span></td>
        <td><span id="unitTemperature"></span></td>
        <td><span id="power"></span></td>
        <td><span id="unitPower"></span></td>
        <td><span id="blindpower"></span></td>
        <td><span id="unitBlindpower"></span></td>
        <td><span id="rotationspeed"></span></td>
        <td><span id="unitRotationspeed"></span></td>
        <td><span id="bladeposition"></span></td>
        <td><span id="unitBladeposition"></span></td>
    </tr>
</table>
</body>
</html>
```

In der consumer.html-Datei wird mit Hilfe von JavaScript die Daten geholt und in eine HTML-Tabelle eingebettet.
Dann muss in der WindengineController-Klasse der Link entsprechen angepasst werden.

```
public String windengineMain() {
    String mainPage = "This is the windengine application! (DEZSYS_GK72_WINDPARK) <br/><br/>" +
            "<a href='http://localhost:8080/windengine/001/xml'>Link to windengine/001/xml</a><br/>" +
            "<a href='http://localhost:8080/windengine/001/json'>Link to windengine/001/json</a><br/>" +
            "<a href='consumer.html'>See data</a><br/>";
    return mainPage;
```

Nachdem man diese Schritte ausgeführt hat muss man wieder `mvn spring-boot:run` ausgeführt werden und die Seite http://localhost:8080 aufgerufen werden.



Wenn man dann auf den Link: See data klickt sieht man:

ID	Time	Windspeed	Unit Windspeed	Temperature	Unit Temperature	Power	Unit Power	Blindpower	Unit Blindpower	Rotationspeed	Unit Rotationspeed	Bladeposition	Unit Bladeposition
001	2020-11-06 12:03:21.454	77.06	kmH	-12.89	C	90.67	kwH	141.24	kwH	63.87	uM	12	grad

Die Tabelle mit den Daten.

Das Projekt lässt sich unter folgendem Link finden: https://github.com/TGM-HIT/syt4-gk771-windpark-rest-btimmelmayer