GK 771 Middleware Engineering "Marathon - REST"

Spring Boot:

Spring Boot ist ein Framework für Java, dass sehr oft für Web-Anwendungen verwendet wird. Außerdem hilft Spring Boot beim Aufbau von Applikationen, da es sich den Klassenpfad anschaut und die fehlenden Elemente hinzufügt. Spring Boot generiert keinen Code und nimmt auch keine Änderungen an den Dateien vor. Es verbindet beim Starten der Anwendung Beans und Einstellungen und wendet sie auf den Kontext an.

Code:

```
implementation 'com.fasterxml.jackson.dataformat:jackson-dataformat-xml'
```

In der build.gradle-Datei implementiere ich bei den Dependencies, dass xml als Format verwendet werden kann.

```
import org.springframework.http.MediaType;
```

Im Controller habe ich das MediaType-Package importiert, weil sonst produces einen Fehler wirft.

```
"<a href='http://localhost:8080/timingstation/001/xml'>Link to timingstation/001/xml</a><br/>" +
```

Hier habe ich eine weitere Unterseite für die Website erstellt.

```
"<a href='http://localhost:8080/timingstation/001/consumer'>Link to timingstation/001/consumer</a><br/>" +
```

Außerdem habe ich auch noch eine weitere Website für die HTML-Tabelle erstellt.

```
@RequestMapping(value="/timingstation/{timingstationID}/data", produces = MediaType.APPLICATION_JSON_VALUE)
public TimingstationData timingstationData( @PathVariable String timingstationID ) {
    return service.getTimingstationData( timingstationID );
}
```

Das ist die Methode die die Daten im JSON-Format darstellt.

```
@RequestMapping(value="/timingstation/{timingstationID}/xml",  produces = MediaType.APPLICATION_XML_VALUE)
public TimingstationData timingstationXML( @PathVariable String timingstationID ){
    return service.getTimingstationData( timingstationID );
}
```

Das ist die Methode die die Daten im XML-Format darstellt. Das produces-Attribut gibt an in welchem Format es erzeugt werden soll. In diesem Fall XML.

```
@RequestMapping(value="/timingstation/{timingstationID}/consumer",  produces = MediaType.TEXT_HTML_VALUE)
public String timingstationConsumer( @PathVariable String timingstationID ){
    return service.getHTMLData( timingstationID );
}
```

Das ist die Methode die die Daten als HTML-Tabelle darstellen.

```java
public static class CompetitionData {

    private String unitTiming;
    private Party[] party;

    public CompetitionData() {
        this.unitTiming = "hh:mm:ss";
        this.party = new Party[0];
    }

    public String getUnitTiming(){
        return this.unitTiming;
    }

    public Party[] getParty(){
        return this.party;
    }

    public void setParty(Party[] party){
        this.party = party;
    }
```

Ich habe in der Klasse TimingstationData eine innere Klasse mit dem Namen CompetitionData erstellt. Diese ist dazu da um die Daten von den einzelnen Partys zu verwalten.

```java
public static class Party {

    private int partyID;
    private String timing;

    public Party() {
        this.partyID = 0;
        this.timing = "";
    }

    public int getPartyID(){
        return this.partyID;
    }

    public void setPartyID(int partyID){
        this.partyID = partyID;
    }

    public String getTiming(){
        return this.timing;
    }

    public void setTiming(String timing){
        this.timing += timing;
    }

}
```

In der inneren Klasse CompetitionData habe ich eine weitere innere Klasse erstellt mit dem Namen Party. Diese speichert das Timing und die partyID der einzelnen Teilnehmer.

```java
public TimingstationData getData( String inTimingstationID ) {

    TimingstationData data = new TimingstationData();
    data.setTimingstationID( inTimingstationID );
    data.setDistance( 1 );
    data.setAltitude( 200 );
    int anzahl = getRandomInt(1,4);
    TimingstationData.CompetitionData comp = new TimingstationData.CompetitionData();
    TimingstationData.CompetitionData.Party[] party = new TimingstationData.CompetitionData.Party[anzahl];
    for(int partys = 0; partys < anzahl; partys++) {
        party[partys] = new TimingstationData.CompetitionData.Party();
        party[partys].setPartyID(getRandomInt(1, 3000));
        String timing="";
        int stunden = getRandomInt(0, 10);
        int minuten = getRandomInt(0, 59);
        int sekunden = getRandomInt(0, 59);
        if(stunden<10){
            timing += "0"+stunden+":";
        }else{
            timing += stunden+":";
        }
        if(minuten<10){
            timing += "0"+minuten+":";
        }else{
            timing += minuten+":";
        }
        if(sekunden<10){
            timing += "0"+sekunden;
        }else{
            timing += sekunden;
        }
        party[partys].setTiming(timing);
        comp.setParty(party);
    }
    data.setCompetition(comp);
    return data;
}
```

Die Methode getData in der Klasse TimingstationSimulation habe ich so angepasst, dass sie die Daten richtig speichert.

```java
public String htmlDaten( TimingstationData td ){
    TimingstationData data = td;
    String id = data.getTimingstationID();
    String time = data.getTimestamp();
    String daten="<!DOCTYPE html><html><style>table, th, td {border:1px solid black;}</style><body>";
    daten += "<p>TimingstationID: "+id+", Timestamp: "+time+", Distance: 1.0, Altitude: 200.0, unitTiming: hh:mm:ss</p>";
    daten += "<h2>Partys Tabelle</h2>";
    daten += "<table style=\"width:50%\"><tr><th>PartyID</th><th>Timing</th></tr>";
    TimingstationData.CompetitionData comp = td.getCompetition();
    TimingstationData.CompetitionData.Party[] party = comp.getParty();
    int laenge = party.length;
    for(int i=0; i<laenge; i++){
        int partyID = party[i].getPartyID();
        String timing = party[i].getTiming();
        daten += "<tr><td>"+partyID+"</td><td>"+timing+"</td></tr>";
    }
    return daten;
}
}
```

Die Methode htmlDaten in der Klasse TimingstationSimulation erzeugt das String mit den nötigen Daten.


```
{"timingstationID":"001","timestamp":"2021-10-19 14:12:00.459","distance":1.0,"altitude":200.0,"competition":{"unitTiming":"hh:mm:ss","party":[{"partyID":1272,"timing":"03:31:20"},
{"partyID":1870,"timing":"08:11:48"},{"partyID":136,"timing":"10:54:52"}]}}
```


Das ist die JSON-Ausgabe der Timingstation.

Das ist die XML-Ausgabe der Timingstation.

TimingstationID: 001, Timestamp: 2021-11-16 09:30:35.987, Distance: 1.0, Altitude: 200.0, unitTiming: hh:mm:ss

Partys Tabelle

PartyID	Timing
2096	00:17:60
732	00:25:42
146	00:22:54
1215	00:08:13
2579	00:18:02
55	00:27:54
1717	00:47:12
1294	00:52:36

Das ist die HTML-Ausgabe der Timingstation.