

Fehlerhafter Code für das Einlesen der Csv files

Driver

```
private void readTeamsAndDriversFromFile(String teamFileName) {
    ClassLoader classLoader = getClass().getClassLoader();
    File file = new File(classLoader.getResource(teamFileName).getFile());

    try(Scanner scanner = new Scanner(file)){
        scanner.nextLine();
        while (scanner.hasNextLine()){
            String[] param = scanner.nextLine().split(";");
            persistTeamAndDrivers(param);
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}
```

Races

```

private void readRacesFromFile(String racesFileName) {
    ClassLoader classLoader = getClass().getClassLoader();
    File file = new File(classLoader.getResource(racesFileName).getFile());

    try(Scanner scanner = new Scanner(file)){
        scanner.nextLine();
        while (scanner.hasNextLine()){
            String[] param = scanner.nextLine().split(";");

            DateTimeFormatter formatter = DateTimeFormatter.ofPattern(
"dd.MM.yyyy");

            LocalDate dateTime = LocalDate.parse(param[2], formatter);
            Race race = new Race(Long.parseLong(param[0]),param[1],dateTime);
            if(race != null){
                em.persist(race);
            }
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}

```

Bei diesem Code Segmenten funktionierte das Einlesen der Csv Files nicht. ## Funktionaler Code für das einlesen der Csv files == Driver

```

private void readTeamsAndDriversFromFile(String teamFileName) {
    ClassLoader classLoader = getClass().getClassLoader();
    File file = new File(classLoader.getResource(teamFileName).getFile());

    try(Scanner scanner = new Scanner(file, "UTF-8")){
        scanner.nextLine();
        while (scanner.hasNextLine()){
            String[] param = scanner.nextLine().split(";");
            persistTeamAndDrivers(param);
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}

```

Races

```

private void readRacesFromFile(String racesFileName) {
    ClassLoader classLoader = getClass().getClassLoader();
    File file = new File(classLoader.getResource(racesFileName).getFile());

    try(Scanner scanner = new Scanner(file, "UTF-8")){
        scanner.nextLine();
        while (scanner.hasNextLine()){
            String[] param = scanner.nextLine().split(";");

            DateTimeFormatter formatter = DateTimeFormatter.ofPattern(
"dd.MM.yyyy");

            LocalDate dateTime = LocalDate.parse(param[2], formatter);
            Race race = new Race(Long.parseLong(param[0]),param[1],dateTime);
            if(race != null){
                em.persist(race);
            }
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}

```

Aufgrund der fehlenden UTF-8 Annotation konnten meine Files in Windows und Linux nicht eingelesen werden. Durch ergänzen von "UTF-8" beim Scanner, lassen sich die Files einlesen.

Checken ob der Teamname schon in der Tabelle existiert

Nicht funktionierender Code

```

private void persistTeamAndDrivers(String[] line) {
    if(em.createNamedQuery("Team.findExistingTeam", Team.class).setParameter("NAME",line[0]).getSingleResult() == null){
        Team team = new Team(line[0]);
        em.persist(team);
        em.persist(new Driver(line[1],team));
        em.persist(new Driver(line[2],team));
    }
}

```

Es kam die Exception 'NoResultException: No entity found for query', da ja auch noch keine Teamnamen in der Tabelle standen.

Funktionierender Code

```
private void persistTeamAndDrivers(String[] line) {
    Team team = null;
    try {
        team = new Team();
        team = em.createNamedQuery("Team.findExistingTeam", Team.class)
            .setParameter("NAME", line[0])
            .getSingleResult();
    } catch (NoResultException e) {
        team = new Team(line[0]);
        em.persist(team);
    }
    em.persist(new Driver(line[1], team));
    em.persist(new Driver(line[2], team));
}
```

RestConfig

```
@ApplicationPath("api")
public class RestConfig extends Application {
}
```

RestConfig Klasse erstellt

getSumPointsOfDrivers()

Ich hatte diese Methode während des Tests noch nicht ausprogrammiert. Ebenfalls hatte ich die Query noch nicht geschrieben.

```
@GET
public JsonObject getPointsSumOfDriver(@QueryParam("name") String name) {

    long sumPoints = em.createNamedQuery("Result.sumPointsForAllDrivers", Long
.class)
        .setParameter("NAME", name)
        .getSingleResult();
    JsonObject jsonObject = Json.createObjectBuilder()
        .add("driver", name)
        .add("points", sumPoints)
        .build();
    return jsonObject;
}
```

```
@NamedQueries({
    @NamedQuery(name = "Result.sumPointsForAllDrivers", query = "select
sum(r.points) from Result r where r.driver = (select d.id from Driver d where d.name
= :NAME)")
})
```

getWinnerOfCountry()

fehlerhafter Code

```
public Response findWinnerOfRace() {
    Race winnerOfCountry = em.createNamedQuery("Result.findWinnerOfCountry", Result
.class).setParameter();*/
    return Response.ok(driverName).build();
}
```

Die Methode habe ich zeitlich nicht ausprogrammieren können

```
@GET
@Path("winner/{country}")
@Produces(MediaType.APPLICATION_JSON)
public Response findWinnerOfRace(@PathParam("country") String country) {

    Long driverId = em.createNamedQuery("Result.getWinnerOfRace", Driver.class)
        .setParameter("COUNTRY", country)
        .getSingleResult().getId();

    Driver driverName = em.find(Driver.class, driverId);
    return Response.ok(driverName).build();
}
```

racessWonByTeams()

Ich hatte ebenfalls keine Zeit um diese Methode während des Tests auszuprogrammieren

```

@GET
@Path("raceswon")
@Produces(MediaType.APPLICATION_JSON)
public List<Race> racesWonByTeams(@QueryParam("team") String team){
    List<Race> races = em.createNamedQuery("Race.racesWonByTeams",Race.class)
        .setParameter("TEAM",team)
        .getResultList();

    return races;
}

```

```

@NamedQuery(name = "Race.racesWonByTeams",query = "select re.race from Result re where
re.position = 1 and re.driver in (select distinct d.id from Driver d where d.team =
(select t.id from Team t where t.name like :TEAM))")

```

allRacesWonByTeam()

Diese Aufgabe war zwar nur eine Spezialistenaufgabe ich wollte sie aber trotzdem probieren und verstehen können.

```

@GET
@Produces(MediaType.APPLICATION_JSON)
@Path("all")
public List<String[]> allRacesWonByTeam(){
    List<Driver> drivers = em.createNamedQuery("Driver.findAll",Driver.class)
        .getResultList();
    List<String[]> driverAndPoints = new LinkedList<>();

    for (Driver driver: drivers) {
        Long points = em.createNamedQuery("Result.getAllPoints",Long.class)
            .setParameter("DRIVER",driver)
            .getSingleResult();
        driverAndPoints.add(new String[]{driver.toString(), " " +points});
    }
    return driverAndPoints;
}

```

```

@NamedQuery(name = "Driver.findAll", query = "select d from Driver d")
@NamedQuery(name = "Result.getAllPoints",query = "select sum(r.points) from Result r
where r.driver = :DRIVER")

```