RefereeManager

Release 0.18-SNAPSHOT

12.04.2021

Inhalte:

Schiedsrichterverwaltung für Tischtennis

KAPITEL 1

RefereeManager – Das Nutzerhandbuch

Schiedsrichterverwaltung für Tischtennis

1.1 Einleitung

Was soll das und wie kann ich mitmachen?

Der *RefereeManager* ist eine Schiedsrichterverwaltung für Tischtennisschiedsrichterinnen.

Dass andere Sportarten nicht explizit unterstützt werden, schließt nicht aus, dass der *RefereeManager* in Teilbereichen nicht doch nützlich sein kann. Wenn man die angebotenen Features nutzen möchte – voilà.

Der RefereeManager ist ein kostenloses, freies und quelloffenes (Open Source) Programm. Das wird auch so bleiben, eine Kommerzialisierung ist nicht geplant und wird nicht stattfinden. Die Quellen liegen unter https://gitlab.com/open-tt/refereemanager

1.1.1 Features

Folgende Features bietet der RefereeManager (bzw. wird bieten):

1.0.0 – Meilenstein Basics

- **Verwaltung von Personen (Schiedsrichterinnen, Vereinsverantwortliche, Berichtsempfängerinnen, Auszubildende, . . .) un**
  - Name und Titel
  - Adressen

    - Aus- und Fortbildung
    - Status
    - Geburtstage, Todestage
- Verwaltung von Clubs
- Verwaltung von Ligen
- Verwaltung von zusätzlich notwendigen Daten (Geschlechter, Status, Ausbildungsarten, . . .)
- Kommunikation per E-Mail mit Personen
- Erzeugung von Dokumenten oder Listen aus den Daten

2.0.0 – Meilenstein Dates, Assignments, Teams

- Verwaltung von Terminen
- Verwaltung von Schiedsrichtereinsätzen
- Verwaltung von Turniereinsätzen
- Verwaltung von Teams

3.0.0 – Meilenstein Refinement

- andere Sprachen
- Import von Daten anderer Systeme
- Bereinigung der Datenbasis
- Behandlung der Daten ausgeschiedener Personen

1.1.2 Daten

Die Daten des *RefereeManager* werden in XML-Dateien lokal auf dem Rechner abgelegt. Pro Saison wird eine XML-Datei erstellt.

Das heißt, für die Nutzung muss man nicht online sein und für die Datensicherung ist jede Nutzerin selbst verantwortlich. Diese Entscheidung wurde bewusst getroffen, es ist auch nicht geplant, eine Onlinespeicherung zu implementieren.

Das heißt ebenfalls, dass die meisten Daten der Personen redundant pro Saison gespeichert werden. Diese Entscheidung wurde ebenfalls bewusst getroffen, da sonst die Datenmodellierung insbesondere des Archivs deutlich komplizierter geworden wäre. So müssen Daten kaum historisiert werden.

1.1.3 Mitmachen

Wie erwähnt, ist der *RefereeManager* Open Source, das heißt, jede Nutzerin oder Programmiererin kann sich am Projekt beteiligen oder ihre eigene Version des Programms erstellen.

Das Programm wird von mir (Ekkart Kleinod) in meiner Freizeit entwickelt, das heißt, ich bin frei in der Richtung, die ich dem Programm geben möchte. Das heißt auch, ich programmiere, wenn ich Zeit und Lust dazu habe. Interessante Features werde ich umsetzen, wenn sie mir nützen oder eine technische Herausforderung für mich darstellen.

Ergo: wenn etwas schnell umgesetzt werden soll: begeistert mich oder macht das selbst und erstellt einen Pull-Request. Was nicht hilft (ernsthaft): mir Geld anbieten.

Wenn Ihr Geld ausgeben wollt, sucht Euch eine Programmiererin, die Euer Feature umsetzt und als Pull-Request einreicht.

Wobei könnt Ihr helfen?

- Fehler melden
- Verbesserungen und Wünsche melden
- Übersetzungen in andere Sprachen (ab Meilenstein 3.0.0)
- selbst programmieren

Der Programmcode, Fehler, Wünsche und die Dokumentation werden über **gitlab** gehostet. Ihr besorgt Euch also am besten einen Account dafür.

Quellcode per git in gitlab unter: https://gitlab.com/open-tt/refereemanager

Fehler melden in gitlab unter: https://gitlab.com/open-tt/refereemanager/issues

Wünsche melden in gitlab unter: https://gitlab.com/open-tt/refereemanager/issues

Übersetzungen gebe ich bekannt, wenn es soweit ist, derzeit ist die technische Basis dafür noch nicht da

Siehe auch: *RefereeManager – The Developer Manual*

1.2 Installation

Derzeit mit der Holzhammermethode:

- Java 12 des OpenJDK installieren
- neueste jar-Datei des *RefereeManager* herunterladen
- jar-Datei mit Java aufrufen `java -jar refereemanager.jar` bzw. `java -jar refereemanager-pre.jar`

Downloads:

- OpenJDK
- RefereeManager

Zu tun: Installationsanleitung verbessern

Der *RefereeManager* ist eine Schiedsrichterverwaltung für Tischtennisschiedsrichterinnen, von VSR (Verbandsschiedsrichterin) bis IU (International Umpire).

Andere Sportarten werden nicht explizit unterstützt, evtl. kann die Software aber auch dafür eingesetzt werden.

Die Anrede in diesem Handbuch ist durchgängig weiblich. Personen anderen Geschlechts können natürlich verwaltet werden bzw. das Programm bedienen.

Bemerkung: Das Programm ist noch stark in der Entwicklung.

Bis zur Version 1 kann sich insbesondere das Datenmodell noch ändern, ohne dass eine automatische Datenübernahme angeboten wird.

KAPITEL 2

RefereeManager – The Developer Manual

Referee management for table tennis

2.1 Structure, Code, and Files

Links and workflows.

2.1.1 Structure

The repository is structured as follows:

`build` build files for releases

`documentation` documentation, especially user, developer, and API documentation

`files` released files

`refereemanager` eclipse project with sources, ressources and tests

`submodules` needed submodules (edgeutils for Ant and JAXB commons‘

2.1.2 Git-Repository

The **`git`** repository is maintained using **`gitlab`**. **`gitlab`** is used for code maintenance and issue tracking.

Repository https://gitlab.com/open-tt/refereemanager/

Issue Tracker https://gitlab.com/open-tt/refereemanager/issues

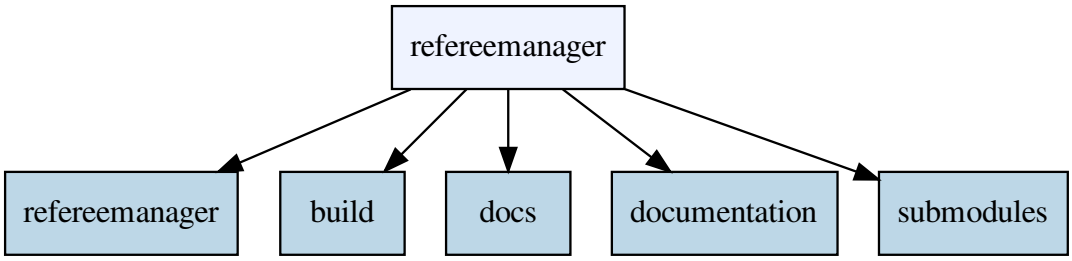

Abb. 1: Repository Structure.

The branching model regards to the stable mainline model described in http://www.bitsnbites.eu/a-stable-mainline-branching-model-for-git/.

This means, there is always a stable mainline, the `master` branch. This branch ist always compileable and testable, both without errors.

Features are developed using `feature` branches. Special feature branches are used (different from the mainline model) for finalizing releases.

Releases are created as branches of the mainline. Additionally, each release is tagged, tags contain the patch and, if needed, additional identifiers, such as `rc1` or `beta`.

Patches are made in the according release branch. Minor version changes get their own release branch.

Naming Feature Branches (with ticket)

- `feature/t`*`ticket number-description`*
- `feature/t43-stable_mainline`
- `feature/t39-use_kotlin`

Naming Feature Branches (without ticket)

- `feature/`*`description`*
- `feature/documentation`

Naming Feature Branches for Finalizing Releases

- `feature/release-`*`major.minor.patch`*`[-`*`additional`*`]`
- `feature/release-0.15.0`
- `feature/release-0.17.0-pre1`

Naming Release Branches

- `release/major.minor`
- `release/0.15`
- `release/1.0`

Naming Tags

- `major.minor.patch[-additional]`
- `0.15.0`
- `1.0.0-beta`
- `1.0.0-rc1`
- `1.0.0`

2.1.3 Releases

See also: *Create a release*

Releases are tagged in the **`git`** repository:

- https://gitlab.com/open-tt/refereemanager/tags

The according binaries are maintained at **`sourceforge`**:

- https://sourceforge.net/projects/refereemanager/
- https://sourceforge.net/projects/refereemanager/files/

2.2 Setup your working environment

Get working...

2.2.1 Software

What is needed for development?

I develop under **`LinuxMint`** (**`Ubuntu`**) and **`Windows`**. Installer creation for **`Windows`** can be done in **`Windows`** and **`LinuxMint`**, for **`Linux`** (**`Ubuntu`**) in **`LinuxMint`** only.

Mandatory Software

Java

The RefereeManager is written in **`Java 9+`**, using **`JavaFX`**, and **`JAXB`**. It uses language features of the newest version available in LinuxMint and supported by all maven plugins.

You can check the current Java version in `pom.xml` of project `refereemanager`.

In https://gitlab.com/open-tt/refereemanager/blob/master/refereemanager/pom.xml look for the line

```
<java.version>...</java.version>
```

Git

You need **git** for checking out the code and providing merge requests.

Maven

Maven is used for providing the external modules for the code. If you use **eclipse**, a sufficient maven is mostly included.

Recommended Software

Ant

You need **ant** for execution of the JAXB and release scripts. You can choose not to use ant, in this case you have to execute the required scripts by hand.

Eclipse

The development of the *RefereeManager* takes place in **eclipse**, an eclipse project and launches are provided in the repository. It is possible to use another development environment, I just cannot give any advice outside eclipse.

Software for Creation of Releases

dpkg

The Debian (Ubuntu) installer is created using **dpkg**.

NSIS

The Windows installer is created with **NSIS**. This is very painful and, at the moment, very unstable. I am not sure if I stick with NSIS in the future.

At the moment, the **edgeutils** project has to be installed locally via `maven install`. This will be fixed in the future.

In short, do the following:

1. check out the **refereemanager** and the **edgeutils** repository
2. update/check out the submodules
3. import the **eclipse** projects `de.edgesoft.edgeutils` and `edgeutils-modules`
4. wait for the **maven** dependencies to download and install
5. install **edgeutils** via `maven install` on `edgeutils-modules`
6. import the **eclipse** project `refereemanager`

7. wait for the **maven** dependencies to download and install
8. check, if the **refereemanager** can be compiled and executed
9. start development

refereemanager repository https://gitlab.com/open-tt/refereemanager/

edgeutils repository https://gitlab.com/ekleinod/edgeutils/

2.3 Create a release

How to create a release.

It is assumed, that all development branches (`feature`) are merged into the `master` branch and the release is created from the `master` branch. The examples will use release *0.15.0* as example.

If you are not sure about a step, save your progress in **git**, then execute the step. This way you can check the status with `git status` and `git diff file` and go back to the last step, if needed, with `git checkout -- file`.

1. create a feature branch for finalizing the release (this differs from the mainline model, but I find it easier to work this way)

```
repopath $ git checkout -b feature/release-0.15.0
repopath $
```

2. update `changelog` and `changelog.md`

   The first file is for debian packages, the latter for humans. Maybe in the future there will be a script converting one to the other, but for now redundancy rules.

3. check if `README.md` is up-to-date
4. check if documentation is up-to-date
5. update `refereemanager/src/main/resources/project.properties`
   - version numbers
   - copyright information
6. update version number in `build/build.xml`, as this cannot be done via script
7. goto `build` directory

```
repopath $ cd build/
repopath/build $
```

8. set version and copyright information in all files in directory `refereemanager` via **ant** task

```
repopath/build $ ant setversion

Buildfile: repopath/build/build.xml

clearLog:

setversion:
  [echo] @version
  [echo] docversion
  [echo] copyright
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
  [echo] Ready, now create new jar, edit changelog, then call 'sudo ant debinstall
↪' (as root).

BUILD SUCCESSFUL
Total time: 1 second

repopath/build $
```

9. create new jar
   1. update `refereemanager/pom.xml` with correct version, check other information
   2. in **`eclipse`** right click `pom.xml`, then select *Run As* → *Maven build*

      This calls **`maven`** with the goals **`clean compile package`**. If you do not use **`eclipse`**, call **`maven`** accordingly.
   3. copy the generated jar from `refereemanager/target/refereemanager-`*`version`*`.jar` to `files/refereemanager.jar`
   4. remove generated `refereemanager/dependency-reduced-pom.xml`
   5. remove generated `refereemanager/target/` to get rid of old artifacts for further development
10. create **`deb`** installer files via **`ant`** task

Wichtig: At this point `changelog` and `changelog.md` have to contain the correct release date (and time), as they are used in the generation steps.

The task has to be called as *root* in order for *user* and *group* of the files to be correctly set.

```
repopath/build $ sudo ant debinstall

Buildfile: repopath/build/build.xml

clearLog:

debinstall:
     [copy] Copying 6 files to repopath/build/debian-tmp
     [gzip] Building: repopath/build/debian-tmp/usr/share/doc/refereemanager/
↪changelog.gz
     [echo] Ready, now call 'ant createdeb' (not as root).

BUILD SUCCESSFUL
Total time: 1 second

repopath/build $
```

11. create deb file via **`dpkg`** and **`ant`** task

```
repopath/build $ ant createdeb

Buildfile: repopath/build/build.xml

createdeb:
   [move] Moving 1 file to repopath/files
   [echo] Ready, now call 'sudo ant clearTmp' (as root).
```

(Fortsetzung auf der nächsten Seite)

(Fortsetzung der vorherigen Seite)

```
BUILD SUCCESSFUL
Total time: 5 seconds

repopath/build $
```

12. remove temporary files as *root* via **ant** task

```
repopath/build $ sudo ant clearTmp

Buildfile: repopath/build/build.xml

clearTmp:
 [delete] Deleting directory repopath/build/debian-tmp
   [echo] Ready, now call 'ant wininstall'.

BUILD SUCCESSFUL
Total time: 0 seconds

repopath/build $
```

13. create **windows** installer via **NSIS** and **ant** task

```
repopath/build $ ant wininstall

Buildfile: repopath/build/build.xml

clearLog:

wininstall:
     [copy] Copying 1 file to repopath/build/win-tmp

installer:

nsis:windows:

nsis:unix:

nsis:
     [move] Moving 1 file to repopath/files
   [delete] Deleting directory repopath/build/win-tmp

BUILD SUCCESSFUL
Total time: 1 second

repopath/build $
```

14. check if all files look good
15. merge feature branch to master without fast forward

```
repopath $ git checkout master
...
repopath $ git merge --no-ff feature/release-0.15.0
...
repopath $
```

16. remove feature branch, remote too, if needed

```
repopath $ git branch -d feature/release-0.15.0
...
repopath $ git push origin --delete feature/release-0.15.0
...
repopath $
```

17. create release branch, tag release

```
repopath $ git checkout -b release/0.15
...
repopath $ git tag -a 0.15.0
...
repopath $
```

18. push everything if you haven't done so yet

```
repopath $ git push --all
...
repopath $ git push --tags
...
repopath $
```

19. upload files to sourceforge
    - https://sourceforge.net/projects/refereemanager/files/

2.4 Implementation details

Here you find some details/tips about/for the implementation.

2.4.1 Create a new Overview

An overview contains a table on the left, a detail view on the right, and below the detail view the buttons for editing the data.

An example is the *referee overview*, visible using menu *Menschen* → *Schiriübersicht*, or `Control-Alt-R`.

For a new overview you have to adapt/create at least the following files (example overview for *trainees*, all files relative to the `de.edgesoft.refereemanager` package):

Adapt

- `view.AppLayout.fxml`
- `controller.AppLayoutController.java`

Overview and Details views

- `view.datatables.DataTableTrainees.fxml`
- `controller.datatables.DataTableTraineesController.java`
- `view.details.DetailsTrainee.fxml`
- `controller.details.DetailsTraineeController.java`

- `controller.overview.OverviewTraineesController.java`

Editing dialog

- `view.editdialogs.EditDialogTrainee.fxml`
- `controller.editdialogs.EditDialogTraineeController.java`
- `view.inputforms.InputForm`*xyz*`.fxml` if needed
- `controller.inputforms.InputForm`*xyz*`Controller.java` if needed

The developer manual is written in English, because I program in English. As I am no native speaker, some phrases might sound German.

> **Warnung:** I switched to Java 9+ recently, the documentation does not reflect all changes yet.

Source code per git in gitlab unter: https://gitlab.com/open-tt/refereemanager

Fehler melden in gitlab unter: https://gitlab.com/open-tt/refereemanager/issues

Wünsche melden in gitlab unter: https://gitlab.com/open-tt/refereemanager/issues

Übersetzungen gebe ich bekannt, wenn es soweit ist, derzeit ist die technische Basis dafür noch nicht da

KAPITEL 3

ToDo-Liste

Zu tun: Installationsanleitung verbessern

(Der ursprüngliche Eintrag steht in /home/docs/checkouts/readthedocs.org/user_builds/refereemanager/checkouts/latest/docs/source/user
Zeile 15.)

Der *RefereeManager* ist eine Schiedsrichterverwaltung für Tischtennisschiedsrichter, von VSR bis IU.

Andere Sportarten werden nicht explizit unterstützt, evtl. kann die Software aber auch dafür eingesetzt werden.

Die Anrede in diesem Handbuch ist durchgängig weiblich. Personen anderen Geschlechts können natürlich verwaltet werden bzw. das Programm bedienen.

Bemerkung: Das Programm ist noch stark in der Entwicklung.

Bis zur Version 1 kann sich insbesondere das Datenmodell noch ändern, ohne dass eine automatische Datenübernahme angeboten wird.

Stichwortverzeichnis