Bachelor Technische Informatica

Kroket

Kroket

Detailed Design Document

Authors:
Peter van Heck (0649886)
Peter Koymans (0748876)
Kay Lukas (0758084)
Astrid Pieterse (0743073)
Robbert Raats (0675661)
Willem Sonke (0752229)
Roby Visser (0654428)

Project Manager:
Sebastiaan Candel

Quality Assurance Manager:
Ronald van Zon

Senior Management:
Mark van den Brand, MF 7.096
Lou Somers, MF 7.145

Advisor:
Erik Scheffers, MF 7.146

Customer:
Lex Lemmens, HG -1.27

October 26, 2012 Eindhoven
Abstract

This document contains a detailed description of all code in the kroket application. This application is developed for the Software Engineering Project (2IP35) at Eindhoven University of Technology.

The document complies with the Detailed Design Document (DDD) from the Software Engineering Standard, as set by the European Space Agency.
## Contents

1. **HTML**  
   1.1 help.html ................................. 8  
   1.2 about.html ............................. 9  
   1.3 index.html ............................ 9  

2. **CSS**  
   2.1 layout.css ............................ 16  
   2.2 printlayout.css ...................... 29  

3. **JavaScript**  
   3.1 studyplanner.js ...................... 31  
   3.2 searchSubjects.js .................... 41  
   3.3 searchPackages.js ................... 44  
   3.4 load.js ............................... 46  
   3.5 printversion.js ...................... 49  
   3.6 authentication.js ................... 50  
   3.7 storage.js ............................ 54  
   3.8 persistentData.js ................... 58  
   3.9 validator.js .......................... 61  
   3.10 urls.js .............................. 64  
   3.11 translation.js ...................... 68  
   3.12 translation-nl_NL.js ............... 69  
   3.13 utils.js ............................. 70  

4. **Python (Django)**  
   4.1 KroketApp.models .................... 79  
   4.2 KroketApp.admin ...................... 103  
   4.3 KroketApp.categorize ............... 108  
   4.4 KroketApp.savemodels .............. 116  
   4.5 KroketApp.urls ....................... 126  
   4.6 KroketApp.recommends ............. 127  
   4.7 KroketApp.security .................. 129
## CONTENTS

### 5 Python (queries)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>KroketApp.query.urls</td>
</tr>
<tr>
<td>5.2</td>
<td>KroketApp.query.major.list</td>
</tr>
<tr>
<td>5.3</td>
<td>KroketApp.query.major.subjects</td>
</tr>
<tr>
<td>5.4</td>
<td>KroketApp.query.package.search</td>
</tr>
<tr>
<td>5.5</td>
<td>KroketApp.query.package.subjects</td>
</tr>
<tr>
<td>5.6</td>
<td>KroketApp.query.recommendation.subject</td>
</tr>
<tr>
<td>5.7</td>
<td>KroketApp.query.schedule.load</td>
</tr>
<tr>
<td>5.8</td>
<td>KroketApp.query.schedule.save</td>
</tr>
<tr>
<td>5.9</td>
<td>KroketApp.query.schedule.list</td>
</tr>
<tr>
<td>5.10</td>
<td>KroketApp.query.schedule.rename</td>
</tr>
<tr>
<td>5.11</td>
<td>KroketApp.query.schedule.delete</td>
</tr>
<tr>
<td>5.12</td>
<td>KroketApp.query.schedule.validate</td>
</tr>
<tr>
<td>5.13</td>
<td>KroketApp.query.subject.info</td>
</tr>
<tr>
<td>5.14</td>
<td>KroketApp.query.subject.search</td>
</tr>
<tr>
<td>5.15</td>
<td>KroketApp.query.user.authenticate</td>
</tr>
<tr>
<td>5.16</td>
<td>KroketApp.query.user.change</td>
</tr>
</tbody>
</table>

### 6 Python (jobs)

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>KroketApp.schedule.daily_jobs</td>
</tr>
<tr>
<td>6.2</td>
<td>KroketApp.schedule.daily_jobs.subject_updater.addData</td>
</tr>
<tr>
<td>6.3</td>
<td>KroketApp.schedule.daily_jobs.subject_updater.biztalk</td>
</tr>
<tr>
<td>6.4</td>
<td>KroketApp.schedule.daily_jobs.subject_updater.parser</td>
</tr>
<tr>
<td>6.5</td>
<td>KroketApp.schedule.daily_jobs.recommender_job</td>
</tr>
</tbody>
</table>

### 7 Unit tests

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1</td>
<td>test/base-test.js</td>
</tr>
<tr>
<td>7.2</td>
<td>test/studyplanner-test.js</td>
</tr>
<tr>
<td>7.3</td>
<td>test/load-test.js</td>
</tr>
<tr>
<td>7.4</td>
<td>test/authentication-test.js</td>
</tr>
<tr>
<td>7.5</td>
<td>test/validator-test.js</td>
</tr>
<tr>
<td>7.6</td>
<td>test/utils-test.js</td>
</tr>
<tr>
<td>7.7</td>
<td>test/storage-test.js</td>
</tr>
<tr>
<td>7.8</td>
<td>test/searchSubjects-test.js</td>
</tr>
<tr>
<td>7.9</td>
<td>test/searchPackages-test.js</td>
</tr>
<tr>
<td>7.10</td>
<td>KroketApp.tests.addData_tester</td>
</tr>
<tr>
<td>7.11</td>
<td>KroketApp.tests.recommender_job_tester</td>
</tr>
<tr>
<td>7.12</td>
<td>KroketApp.tests.testCategorize</td>
</tr>
<tr>
<td>7.13</td>
<td>KroketApp.tests.testSecurity</td>
</tr>
<tr>
<td>7.14</td>
<td>KroketApp.tests.query.major.testMajor</td>
</tr>
<tr>
<td>7.15</td>
<td>KroketApp.tests.query.package.testPackage</td>
</tr>
<tr>
<td>7.16</td>
<td>KroketApp.tests.query.recommendation.testRecommendation</td>
</tr>
<tr>
<td>7.17</td>
<td>KroketApp.tests.query.schedule.testSchedule</td>
</tr>
<tr>
<td>7.18</td>
<td>KroketApp.tests.query.schedule.testValidate</td>
</tr>
<tr>
<td>7.19</td>
<td>KroketApp.tests.query.subject.subjectTestSet</td>
</tr>
</tbody>
</table>
CONTENTS

7.20 KroketApp.tests.query.subject.testInfo ........................................ 235
7.21 KroketApp.tests.query.subject.testSearch .................................... 242
7.22 KroketApp.tests.query.user.testAuthenticate .............................. 250
7.23 KroketApp.tests.query.user.testChange ...................................... 257

8 Index ......................................................................................... 260
Introduction

This document contains detailed documentation for all code of kroket. Since the application is written in multiple programming languages, the documentation is divided in seven chapters.

- HTML – documentation for the HTML files and the components in it
- CSS – documentation for all CSS rules used for styling the website
- JavaScript – documentation for all JavaScript files
- Python (Django) – documentation for the Django models
- Python (queries) – documentation for the server queries
- Python (jobs) – documentation for the server jobs
- Unit tests – documentation for the unit tests, for both JavaScript and Python

At the beginning of the documentation for a file, a short file description and an index of all classes, functions or rules in this file is placed (the exact contents of this index vary for every language used; for Python the index is hierarchical).

Element documentation

A section is used to document every class, function or rule. Such a section contains the following elements.

Declaration  File name and line number of the declaration
the exact header of the declaration

Qualified name  The fully qualified name, i.e. the name including the package name. Only for Python.

Description  A description of the declaration.

Deprecated  A description about why the element is deprecated. Only if deprecated.
**Parameters**  A list of the parameters of the function. *Only for JavaScript and Python functions/methods, only if parameters exist.*

**Returns**  A description of the returned value. *Only for JavaScript and Python functions/methods, only if there is a return value.*

**References to other elements**

Sometimes it is useful to include a reference to other elements in some documentation. This is done using the following notation. → the-other-method
Chapter 1

HTML documentation

In this chapter, the components in the HTML code will be documented.

All pages listed below, also have a Dutch version. The Dutch version is however mostly the same, so it is not documented separately.

1.1 help.html

This file contains the HTML of the help page of the TU/e Subject Chooser.

1.1.1 Navigation bar

Declaration  Declared in help.html on line 24:
<div class="navbar navbar-fixed-top">

Description  This is the navigation bar on top of the webpage. It contains a link to the About page (see 1.2) and the TU/e logo, which is a link to www.tue.nl.

1.1.2 Left side menu

Declaration  Declared in help.html on line 55:
<div class="span3 bs-docs-sidebar affix-top" data-spy="affix" data-offset-top="160">

Description  This is the menu at the left side of the website. Clicking one of the menu items will scroll the webpage, such that the according paragraph is shown.

1.1.3 Content

Declaration  Declared in help.html on line 68:
<div class="span9">

Description  This div element contains a number of chapters with the actual help text. It is aligned at the right side of the page, such that it is placed beside the menu.
1.2 about.html

This file contains the HTML for the about page of the application.

1.2.1 Navigation bar

Declaration  Declared in about.html on line 22:
\[
<\text{div class="navbar navbar-fixed-top"}>
\]

Description  This is the navigation bar of the website. Contains a link to the help pages and a link to the site of the TU/e.

1.2.2 About the application

Declaration  Declared in about.html on line 46:
\[
<\text{div class="span8"}>
\]

Description  Contains some information about this application, like the supported browsers.

1.2.3 The Kroket group

Declaration  Declared in about.html on line 53:
\[
<\text{div class="span4"}>
\]

Description  Contains the names of the members of the Kroket-group.

1.3 index.html

This file contains the HTML of the main page of the TU/e subject chooser. It uses many scripts, also documented in this DDD. On this page, a user can search for subjects and plan his study.

1.3.1 Navigation bar

Declaration  Declared in index.html on line 48:
\[
<\text{div class="navbar navbar-fixed-top"}>
\]

Description  This is the navigation bar of the website. Contains a link to the help page, to the about page, to the Dutch pages and a link to the site of the TU/e. It also contains the options menu (see 1.3.2) and the Log in and Register option, or the Logged in as [username] menu.
CHAPTER 1. HTML

1.3.2 Options menu

Declaration  Declared in index.html on line 63:
<li id="options" class="dropdown" >

Description  A dropdown menu with the following options:

- **Load** – Loads a saved schedule.
- **Save** – Save the current schedule.
- **Save as** – Save the schedule with a new name.
- **Print schedule** – Show a printable version of the schedule.
- **Validate schedule**
- **Change major** – Used to change the major. Also shows the major the user has chosen.
- **Fill in major subjects** – Fill in all subjects belonging to the chosen major.

1.3.3 Search part

Declaration  Declared in index.html on line 113:
<div id="searchbar" class="span4" >

Description  This is the left side of the screen, containing the search options and search results. Contains the search subjects tab 1.3.4, the search packages tab 1.3.5 and the container of the search results 1.3.6.

1.3.4 Search subjects tab

Declaration  Declared in index.html on line 122:
<div class="tab-pane fade active in" id="subjects-panel" >

Description  This is the pane with the options for searching subjects. It contains a form with the following options:

- **subject-name** – This is the search field, where the user can enter a search term.
- **all-subjects-button**, **deepening-button**, **broadening-button** – A button group of which exactly one button is always pressed. The user can choose to search for deepening subjects, broadening subjects or both.
- **search-options-toggable** – This is the advanced options dropdown, which contains some extra options about the difficulty, year, quartile, timeslot and major of the subjects.

1.3.5 Search packages tab

Declaration  Declared in index.html on line 190:
<div class="tab-pane fade active in" id="coherent-panel">
1.3. INDEX.HTML

Description  This is the panel where the user can search for coherent packages. It contains a few options (listed by their id):

- coherent-name – Here the user can enter a search term.
- coherent-both-button, coherent-use-button, coherent-elective-button – Button group, defines if the application must search for elective packages, USE-packages or both.

1.3.6 Search results container

Declaration  Declared in index.html on line 225:
<id="search-results" class="well well-small" style="max-height: 400px; overflow-y: auto;;">

Description  This is where the search results will be shown. These will be added in this container by the JavaScript of this page, so make sure not to change the id of this container.

1.3.7 Right part

Declaration  Declared in index.html on line 230:
<div class="span8" style="position: relative;">"

Description  This is the right part of the screen, containing the schedule of the user (see 1.3.8), some options, the recommended subjects (see 1.3.9) and also the subject information (see 1.3.10).

1.3.8 Schedule

Declaration  Declared in index.html on line 235:
<div id="schedule-container">

Description  This is where the user’s schedule will be shown. It is an empty container in the HTML, because the table is build by the JavaScript (see doc:js.buildTable). Therefore, the id of this container should not be changed without adapting the JavaScript.

1.3.9 Recommended subjects

Declaration  Declared in index.html on line 248:
<div id='recommended-subjects' class='recommended hide'>

Description  Recommended subjects will be shown inside the div with recommended-subjects-container, which is placed inside this div. This div will be hidden if there are no recommended subjects and will slide down when there are some recommended subjects.


CHAPTER 1. HTML

1.3.10 Subject information

Declaration Declared in index.html on line 256, 257:

```html
<div class="onBottom">
    <div id="subject-info" class = "shadow">

Description This is the window for subject information. It is placed at the bottom of the screen, below the schedule (or partly on top of the schedule if it is too large). It can be collapsed or expanded by the user, by clicking an icon at the top-left corner of this window. It contains five tabs with different information about the subject, which will be described in the following chapters.

1.3.11 General information

Declaration Declared in index.html on line 276:

```html
<div class="tab-pane fade active in" id="subject-info-general-panel">

Description This is a tab inside the Subject information window, which contains some general information about the subject, like the subject name, subject code, number of ects, difficulty level and prerequisites. It also contains a link to the OWINFO page about this subject.

1.3.12 Planning

Declaration Declared in index.html on line 292:

```html
<div class="tab-pane fade in" id="subject-info-planning-panel">

Description This is a tab inside the Subject information window. This tab contains the planning of a subject, shown as the table with id subject-info-planning. This table will be filled by the JavaScript.

1.3.13 Description

Declaration Declared in index.html on line 297:

```html
<div class="tab-pane fade in" id="subject-info-description-panel">

Description This is a tab inside the Subject information window. This tab contains the description of a subject.

1.3.14 Study goals

Declaration Declared in index.html on line 302:

```html
<div class="tab-pane fade in" id="subject-info-studygoal-panel">

Description This is a tab inside the Subject information window. This tab contains the description of a subject.
Description This is a tab inside the Subject information window. This tab contains the study goal of a subject.

### 1.3.15 Evaluation

**Declaration** Declared in index.html on line 307:
```
<div class="tab-pane fade in" id="subject-info-evaluation-panel">
```

Description This is a tab inside the Subject information window. This tab contains the evaluation of a subject. This evaluation is shown as a number inside this tab, in the div with id subject-info-evaluation.

### 1.3.16 Page disabler

**Declaration** Declared in index.html on line 321:
```
<div id="schedule-disabler" style="background-color: rgba(255, 255, 255, 0.6);
top: 0; left: 0; width: 100%; height: 100%; position: absolute; z-index: 1030;
display: none;">
```

Description This is a div that can be shown on top of the entire page. It will make sure no element within the page can be clicked. This is used to disable all options when, for example, the application is loading the schedule of a user. It is transparent, such that you can still see the webpage.

### 1.3.17 Login modal

**Declaration** Declared in index.html on line 331:
```
<div class="modal fade hide" id="loginModal" tabindex="-1" role="dialog" aria-labelledby="loginModalLabel" aria-hidden="true">
```

Description This is the login screen of the application. It is shown when the user clicks Log in in the navigation bar.

### 1.3.18 Start modal

**Declaration** Declared in index.html on line 362:
```
<div class="modal fade hide" id="startModal" tabindex="-1" role="dialog" aria-labelledby="startModalLabel" aria-hidden="true">
```

Description This is the screen shown when a user visits the webpage. It is not always shown, this depends on if the user is already logged in and if his schedule was stored. This modal shows a welcome text to the user. In this screen, the user can choose to login, register himself or continue without login.
CHAPTER 1. HTML

1.3.19 Register modal

Declaration  Declared in index.html on line 396:
<div class="modal fade hide" id="registerModal" tabindex="-1" role="dialog" aria-labelledby="registerModalLabel" aria-hidden="true">

Description  This is a modal shown when the user wants to register himself and clicks Register in the navigation bar.

1.3.20 Alert modal

Declaration  Declared in index.html on line 434:
<div class="modal fade hide" id="alertModal" tabindex="-1" role="dialog" aria-labelledby="alertModalLabel" aria-hidden="true">

Description  This modal is used to show alerts to the user. The text of this modal is set by the JavaScript (see showAlert 3.13.6). It is used instead of the default alert() functionality, because these alerts are styled differently and furthermore, they can (and will) be blocked by some browsers if you use them often.

1.3.21 Major modal

Declaration  Declared in index.html on line 446:
<div class="modal fade hide" id="majorModal" tabindex="-1" role="dialog" aria-labelledby="majorModalLabel" aria-hidden="true">

Description  In this modal the user can fill in his major and the first year of his study. The modal is shown when the user presses Change major inside the Options menu.

1.3.22 Validator modal

Declaration  Declared in index.html on line 485:
<div class="modal fade hide" id="checkModal" tabindex="-1" role="dialog" aria-labelledby="checkModalLabel" aria-hidden="true">

Description  This modal shows the results when the application validates the user’s schedule, which is done when the user clicks Validate schedule.

1.3.23 Save modal

Declaration  Declared in index.html on line 512:
<div class="modal fade hide" id="saveModal" tabindex="-1" role="dialog" aria-labelledby="saveModalLabel" aria-hidden="true">
Description  This modal shows an input field where the user can name his schedule. This is done if the user clicks Save or Save as.

1.3.24  Load modal

Declaration  Declared in index.html on line 537:

```html
<div class="modal fade hide" id="loadModal" tabindex="-1" role="dialog" aria-labelledby="loadModalLabel" aria-hidden="true">
</div>
```

Description  This modal shows a dropdown with the saved schedules of the user, such that he can load one of them. The user can also delete a schedule in this modal. The modal is shown when the user clicks Load.

1.3.25  Overwrite saved schedule modal

Declaration  Declared in index.html on line 563:

```html
<div class="modal fade hide" id="overwriteModal" tabindex="-1" role="dialog" aria-labelledby="overwriteModalLabel" aria-hidden="true">
</div>
```

Description  The modal shows a warning to indicate the user has already saved another schedule with the given name, when the user clicks Save. The user can choose to overwrite this schedule, or to go back to the Save modal.

1.3.26  Log out modal

Declaration  Declared in index.html on line 579:

```html
<div class="modal fade hide" id="logoutModal" tabindex="-1" role="dialog" aria-labelledby="logoutModalLabel" aria-hidden="true">
</div>
```

Description  This modal is shown when the user tries to log out, but has not yet saved his schedule. The user can choose to log out anyway, or to stay logged in.
Chapter 2

CSS documentation

In this chapter, all of the client CSS style sheets will be documented. CSS rules are ordered on the file they are in.

2.1 layout.css

This file contains rules for the general layout of the main page.

Class index  This file contains CSS rules for the following selectors:

```css
#alertModal
#recommended-subjects-container .keyword-highlighting
#schedule-container .table
#searchbar .nav-tabs
#searchbar .tab-content
#subject-info
#subject-info .nav-tabs
#subject-info-header-icon
#subject-info-hideable .tab-pane
.addCourse
.closeWhite
.closeWhite:hover
.conflictingQuartile
.course
.course > div:not(.close)
course.disabledCourse
course.major
course.selectedCourse
course.selectedCourse.major
dropdown-menu .disabledMenuItem
dropdown-menu .disabledMenuItem i
dropdown-menu .disabledMenuItem:hover
```
2.1. LAYOUT.CSS

Detailed documentation for all rules in this file follows.

2.1.1 #schedule-container .table

Declaration Declared in layout.css on line 9:
#schedule-container .table {

Description This rule defines the layout of the schedule on the webpage. It also fixes the widths of the columns and gives the table a minimum width.
CHAPTER 2. CSS

2.1.2 .table

**Declaration**  Declared in *layout.css* on line 17:

```css
.table {
}
```

**Description**  Firefox has a lot of bugs with `border-collapse: collapse`. This should fix this.

2.1.3 .dropdown-menu .disabledMenuItem

**Declaration**  Declared in *layout.css* on line 27:

```css
.dropdown-menu .disabledMenuItem {
}
```

**Description**  A disabled menu item in a dropdown menu.

2.1.4 .dropdown-menu .disabledMenuItem i

**Declaration**  Declared in *layout.css* on line 35:

```css
.dropdown-menu .disabledMenuItem i {
}
```

**Description**  Make the icon lighter (less opaque) for disabled menu items, and don’t make the icon white when hovering a disabled dropdown menu item.

2.1.5 .dropdown-menu .disabledMenuItem:hover

**Declaration**  Declared in *layout.css* on line 43:

```css
.dropdown-menu .disabledMenuItem:hover {
}
```

**Description**  Remove the hover effect for disabled menu items in a dropdown menu.

2.1.6 .tableInnerCell

**Declaration**  Declared in *layout.css* on line 56:

```css
.tableInnerCell {
}
```

**Description**  Class `tableInnerCell` is assigned to all cells of the table where courses could be placed, that is, the complete table except for the *Year* and *Quartile* headers. At the moment, this class does nothing, but it could be extended to support another layout of the table.

2.1.7 .table .subject-code-in-course

**Declaration**  Declared in *layout.css* on line 62:

```css
.table .subject-code-in-course {
}
```

**Description**  This rule hides subject codes of courses that are in the schedule table.
2.1. LAYOUT.CSS

Figure 2.1: A column of the schedule table, containing cells with class `.highlight`. The cells contain `.addCourse` elements, that are shown because of the `.highlight > .addCourse` rule.

2.1.8 `.table-striped tbody tr:nth-child(2n+1) td, .table-striped tbody tr:nth-child(2n+1) th`

Declaration   Declared in layout.css on line 69:
```css
.table-striped tbody tr:nth-child(2n+1) td, .table-striped tbody tr:nth-child(2n+1) th {
```

Description   Change the background colour of Bootstraps striped table rows.

2.1.9 `.table th, .table td`

Declaration   Declared in layout.css on line 76:
```css
.table th, .table td {
```

Description   Set color of the lines that separate the years in the table.

2.1.10 `.highlight`

Declaration   Declared in layout.css on line 87:
```css
.highlight {
```

Description   This rule highlights a cell to show that a course can be placed here with a different background color. See figure 2.1.

2.1.11 `.hoveredHighlight`

Declaration   Declared in layout.css on line 94:
```css
.hoveredHighlight {
```

Description   Used to show to a user that he can drop a subject here, when he drags a subject over a highlighted cell.

2.1.12 `.highlight:hover`

Declaration   Declared in layout.css on line 101:
```css
.highlight:hover {
```
CHAPTER 2. CSS

Figure 2.2: A course with class .course.

Description Sets the cursor to a pointer, to show you can click a highlighted cell to place a course.

2.1.13 .addCourse

Declaration Declared in layout.css on line 113:

```
.course {
}
```

Description This class is given to the “Click or drag here to place” text inside each cell in the schedule, that can be shown to make clear that the user can add courses there. This rule makes the text gray and, in general, less outstanding. Furthermore the opacity is set to 0 (completely transparent). This is overridden in the .highlight > .addCourse rule, to make these elements visible when the .highlight class is added to the containing table cell.

See figure 2.1 for an example of this class.

2.1.14 .highlight > .addCourse

Declaration Declared in layout.css on line 125:

```
.highlight > .addCourse {
}
```

Description Makes addCourse texts visible if they are in a highlight element.

2.1.15 .course

Declaration Declared in layout.css on line 136:

```
.course {
}
```

Description The course class is assigned to every course in the schedule but also in the search results. This class adds a nice layout to courses. Furthermore some properties of Bootstrap .label are overridden. See figure 2.2 for the result.

2.1.16 .course.selectedCourse

Declaration Declared in layout.css on line 155:

```
.course {
}
```

Description Highlights the course that is currently selected, using a different background color and a glow effect. The course is given a slightly higher z-index, to ensure the glow is displayed on top of the other courses.
2.1. LAYOUT.CSS

2.1.17 .course.major

Declaration  Declared in layout.css on line 166:
.course.major {

Description  This rule gives major subjects (.major) a different colour.

2.1.18 .course.selectedCourse.major

Declaration  Declared in layout.css on line 181:
.course.selectedCourse.major {

Description  This rule gives selected major subjects (.major and .selectedCourse) a different colour.

2.1.19 .course.disabledCourse

Declaration  Declared in layout.css on line 190:
.course.disabledCourse {

Description  This rule gives selected major subjects (.disabledCourse) a grey colour. A subject in the search results will be disabled, if that subject is already placed in the schedule.

2.1.20 .course > div:not(.close)

Declaration  Declared in layout.css on line 200:
.course > div:not(.close) {

Description  This rule is applied to all children of a .course, that do not have the .close class. It makes sure the layout of a course does not change when the close icon is added (by reserving some space to place it).

2.1.21 .label

Declaration  Declared in layout.css on line 207:
.label {

Description  Change the .label class from Bootstrap not to have bold text in it.

2.1.22 .package

Declaration  Declared in layout.css on line 218:
.package {


CHAPTER 2. CSS

Figure 2.3: A package containing two subjects. This element is styled by adding the .package class.

Description This rule is applied to every package (.package). This class adds a nice layout to packages. See figure 2.3.

2.1.23 .package:last-child

Declaration Declared in layout.css on line 252:

```
.package:last-child {

Description Removes the bottom margin of the last subject in a package. This is to prevent some empty space below the subject list.

2.1.24 .conflictingQuartile

Declaration Declared in layout.css on line 263:

```

Description Shows a red border and glow around a quartile (table cell) to show that there are two courses in the same timeslot in this quartile. See figure 2.4.

2.1.25 .popover-error .popover-title

Declaration Declared in layout.css on line 273:

```

Description This rule colours the title of error-popovers red. See figure 2.4.

2.1.26 .onBottom

Declaration Declared in layout.css on line 284:

```

Description This is for the Subject information window. This rule anchors the onBottom object to the bottom of the page, and the window will be attached to it.
2.1. LAYOUT.CSS

Figure 2.4: A quartile containing conflicting subjects. The .conflictingQuartile class creates the red border and glow around the quartile. The popover is assigned the .popover-error class such that the .popover-error .popover-title rule colours the title of the popover red.

2.1.27  .recommended

Declaration  Declared in layout.css on line 295:

```css
.recommended {
}
```

Description  This is the container below the schedule, where the recommended subjects can be shown.

2.1.28  .closeWhite

Declaration  Declared in layout.css on line 307:

```css
.closeWhite {
}
```

Description  Makes the remove button of each course white, to improve visibility.

2.1.29  .closeWhite:hover

Declaration  Declared in layout.css on line 316:

```css
.closeWhite:hover {
}
```

Description  Makes the remove button more opaque when it is hovered (with an animation).

2.1.30  #subject-info

Declaration  Declared in layout.css on line 328:

```css
#subject-info {
}
```
CHAPTER 2. CSS

Description Layout of the Subject information window. This rule manages the position of
the window by animating margin-top.

2.1.31 .shadow

Declaration Declared in layout.css on line 343:
.shaddow {

Description Assigns nice rounded borders to the Subject information window, with a black
shadow.

2.1.32 .less-vertical-space

Declaration Declared in layout.css on line 352:
.less-vertical-space {

Description Reduces the vertical space between buttons, that is caused by the default
line-height in Bootstrap.

2.1.33 .keyword-highlighting

Declaration Declared in layout.css on line 359:
.keyword-highlighting {

Description Highlighting of the search term in the text.

2.1.34 #recommended-subjects-container .keyword-highlighting

Declaration Declared in layout.css on line 366:
#recommended-subjects-container .keyword-highlighting {

Description Makes sure keywords are not highlighted when they are actually in a recom-
mended subject and not in the search results.

2.1.35 #searchbar .tab-content

Declaration Declared in layout.css on line 373:
#searchbar .tab-content {

Description Draws a border around the tabbed pane in the searchbar.

2.1.36 #searchbar .nav-tabs

Declaration Declared in layout.css on line 387:
#searchbar .nav-tabs {
Description  A correction for the tabbed pane border in the searchbar.

2.1.37  #subject-info .nav-tabs

Declaration  Declared in layout.css on line 394:
#subject-info .nav-tabs {

Description  A correction for the tabbed pane border in the subject information.

2.1.38  #subject-info-header-icon

Declaration  Declared in layout.css on line 401:
#subject-info-header-icon {

Description  Positioning for the collapse icon in the tab bar of the subject information.

2.1.39  #subject-info-hideable .tab-pane

Declaration  Declared in layout.css on line 410:
#subject-info-hideable .tab-pane {

Description  Enable scrolling in the tabs of the subject information.

2.1.40  .navbar-inner

Declaration  Declared in layout.css on line 420:
.navbar-inner {

Description  Re-color the Bootstrap navbar (the menu bar on the top of the page). See figure 2.5.

2.1.41  .navbar .nav > li > a, .navbar .brand, .navbar .nav > li > a:focus, .navbar .nav > li > a:hover

Declaration  Declared in layout.css on line 434:
.navbar .nav > li > a, .navbar .brand, .navbar .nav > li > a:focus, .navbar .nav > li > a:hover {

Description  Color the text in the navbar white instead of black.

2.1.42  .navbar .nav > li > a:hover

Declaration  Declared in layout.css on line 442:
.navbar .nav > li > a:hover {

25
CHAPTER 2. CSS

Figure 2.5: The styled navbar.
Description  Make hovered text in the navbar grey instead of white.

2.1.43 .navbar .nav > .active > a, .navbar .nav > .active > a:hover, .navbar .nav > .active > a:focus

Declaration  Declared in layout.css on line 449:
.navbar .nav > .active > a, .navbar .nav > .active > a:hover, .navbar .nav > .active > a:focus {

Description  This rule reverts the navbar colouring changes for the active element in the navbar.

2.1.44 .navbar .nav li.dropdown.open > .dropdown-toggle, .navbar .nav li.dropdown.active > .dropdown-toggle, .navbar .nav li.dropdown.open.active > .dropdown-toggle

Declaration  Declared in layout.css on line 463:

Description  This rule reverts the changes for the open element in the navbar.

2.1.45 .navbar-fixed-top .navbar-inner, .navbar-static-top .navbar-inner

Declaration  Declared in layout.css on line 470:
.navbar-fixed-top .navbar-inner, .navbar-static-top .navbar-inner {

Description  A nicer box-shadow for the navbar.

2.1.46 .navbar .brand

Declaration  Declared in layout.css on line 478:
.navbar .brand {

Description  A correction for the TU/e logo in the navbar.

2.1.47 a

Declaration  Declared in layout.css on line 487:
a {

Description  Use the TU/e blue color for the links.
CHAPTER 2. CSS

This subject received the following evaluation by students: 8.3
on a scale from 1 to 10

Figure 2.6: Example of an evaluation in the Subject information window. The element containing “8.3” is styled using .evaluation and .evaluation-8.

2.1.48 .evaluation

Declaration Declared in layout.css on line 502:
      .evaluation {

Description Class for evaluations of subjects. These are labels containing the evaluation, a grade between 1 and 10.

The evaluation-1, evaluation-2, ... classes are used to additionally style an evaluation to have the correct background colour for the evaluation. These rules are omitted from this documentation for brevity.

See figure 2.6 for an example of an evaluation.

2.1.49 #alertModal

Declaration Declared in layout.css on line 525:
      #alertModal {

Description A special Bootstrap modal variant for simple alerts, to avoid having to use document.alert() in JavaScript.

2.1.50 .hiddenSubmit

Declaration Declared in layout.css on line 533:
      .hiddenSubmit {

Description Hides a submit button, is used to make a form submittable by pressing the enter key.
2.2 printlayout.css

This file contains the layout for the print version of the schedule.

Class index  This file contains CSS rules for the following selectors:

\[
\text{.course} \\
\text{.course.major} \\
\text{.table} \\
\text{.table td :first-child} \\
\text{.table td :last-child} \\
\text{.table th} \\
\text{.table th, .table td} \\
\text{.table tr:first-child}
\]

Detailed documentation for all rules in this file follows.

2.2.1 .course

Declaration  Declared in printlayout.css on line 8:
\[
\text{.course} \\
\]

Description  Used for courses in the print version of the schedule (white with a black, dashed border).

2.2.2 .course.major

Declaration  Declared in printlayout.css on line 24:
\[
\text{.course.major} \\
\]

Description  A solid border for major courses.

2.2.3 .table td :first-child

Declaration  Declared in printlayout.css on line 32:
\[
\text{.table td :first-child} \\
\]

Description  Gives the first subject in a quartile nice rounded borders on the top. In this way, all subjects in a quartile form a rectangle with rounded corners.

2.2.4 .table td :last-child

Declaration  Declared in printlayout.css on line 41:
\[
\text{.table td :last-child} \\
\]

Description  Gives the last subject in a quartile nice rounded borders on the bottom.
CHAPTER 2. CSS

2.2.5 .table

Declaration  Declared in printlayout.css on line 49:

```
.table {

```

Description  Used for the schedule in the print version of the schedule.

2.2.6 .table th, .table td

Declaration  Declared in printlayout.css on line 58:

```
.table th, .table td {

```

Description  No black separator lines for the table, the table has no lines.

2.2.7 .table th

Declaration  Declared in printlayout.css on line 65:

```
.table th {

```

Description  Center the text in the table headers.

2.2.8 .table tr:first-child

Declaration  Declared in printlayout.css on line 72:

```
.table tr:first-child {

```

Description  Right-align the text in the first cell of each row (containing Year ..).
Chapter 3

JavaScript documentation

In this chapter, all of the client JavaScript code will be documented.

JavaScript declarations are ordered on the file they are in. Note that in JavaScript, the separation in multiple files is not relevant for the structure of the code. That is, the entire JavaScript code is treated as if it resided in one file. However, the code has been split in several files for readability and manageability.

3.1 studyplanner.js

This is the main JavaScript file for the TU/e Subject Chooser application.

This file contains the most important methods, that don’t easily fit into one of the specialized files for separated components of the application (e.g. authentication or validation). Most of these methods are event-handlers, functions that are directly called, if the user clicks something in the application.

This file also contains the declarations of global variables, except for URLs. Alls used URLs are declared in urls.js.

Variable index  This file contains the following variables:

currentScheduleName ........................................ 34
first ............................................................... 39
isDragging ...................................................... 32
isLoggedIn ....................................................... 33
lockAuthentication ............................................ 32
major ............................................................ 33
majorCourses .................................................. 34
majorName ...................................................... 34
modifiedSchedule ............................................ 32
recommend_timeout .......................................... 40
selectedCourse ............................................... 33
startYear ....................................................... 33
years ........................................................... 33
CHAPTER 3. JAVASCRIPT

Function index  This file contains the following functions:

- CoherentPackage(target_group, name) .................................. 37
- CourseInfo(code, name, ects, level, lastYear, planning, priorknowledge) ........................................ 36
- addYear() ........................................................................ 35
- askMajorCourses() ............................................................ 36
- checkDoubleCourses(cell) .................................................... 39
- clickedCell(event, cell) ....................................................... 38
- clickedCoherentPackage(event, pack) ............................... 37
- clickedCourse(course) .......................................................... 38
- emptySchedule() ................................................................. 40
- fillInCourses(courses, major) .............................................. 35
- fillInSchedule() ................................................................. 35
- recommendSubjects() ............................................................ 40
- removeCourse(clickedicon) .................................................. 38
- removeYear() ..................................................................... 34
- saveMajor() ....................................................................... 36
- setCourses(pack) ................................................................. 38
- setRecommendedSubjects(subjects) ...................................... 40
- setYear(y) .......................................................................... 34
- showMajorInput() ................................................................. 34
- showSubjectInformation(course) .......................................... 39
- updateMajorErrorBar(bool) .................................................. 35
- warning(str, cell) ............................................................... 39

Detailed documentation for all declarations in this file follows.

3.1.1 lockAuthentication

Declaration  Declared in studyplanner.js on line 17:
var lockAuthentication = false;

Description  Will make sure the user can’t post too many login requests in a row.

3.1.2 isDragging

Declaration  Declared in studyplanner.js on line 21:
var isDragging = false;

Description  Boolean indicating whether the user is dragging a subject at the moment.

3.1.3 modifiedSchedule

Declaration  Declared in studyplanner.js on line 25:
var modifiedSchedule = false;
3.1. STUDYPLANNER.JS

Description  Boolean indicating if the schedule has been modified since the last time it was saved.

3.1.4  selectedCourse

Declaration  Declared in studyplanner.js on line 30:
var selectedCourse;

Description  The course that is currently selected by the user.

3.1.5  isLoggedIn

Declaration  Declared in studyplanner.js on line 37:
var isLoggedIn = false;

Description  Whether this user is logged in, according to the client. This could differ from the server-side representation, for example when the session on the server has expired.

3.1.6  years

Declaration  Declared in studyplanner.js on line 45:
var years = 3;

Description  The number of years currently in the table (as an integer).
  This is automatically maintained by addYear (page 35), removeYear (page 34) and buildTable (page 46).

3.1.7  startYear

Declaration  Declared in studyplanner.js on line 54:
var startYear = 2012;

Description  The first year in the table. If this is, for example, 2012, then the first year is 2012-2013. This is given as an integer.
  This variable can be set using setYear (page 34); this method also updates the schedule table with the new year numbers.

3.1.8  major

Declaration  Declared in studyplanner.js on line 62:
var major = -1;
CHAPTER 3. JAVASCRIPT

Description  The user’s major, as an integer. (On the server side, this is called the target_group).

This should not be changed directly, but by using setMajor (page 71), since also menu items should be enabled or disabled, depending on whether a major was chosen or not.

3.1.9  majorName

Declaration  Declared in studyplanner.js on line 67:
var majorName = null;

Description  The name of the user’s major. See also major (page 33).

3.1.10  majorCourses

Declaration  Declared in studyplanner.js on line 73:
var majorCourses;

Description  The courses in this user’s major, as an array of CourseInfo’s (3.1.21).

3.1.11  currentScheduleName

Declaration  Declared in studyplanner.js on line 82:
var currentScheduleName = "";

Description  The name of the currently active schedule. This is used for remembering which schedule should be saved to when the user clicks Save. An empty string means that no schedule name has yet been entered.

This variable is set by saveAsClick (page 54) and used by saveClick (page 54).

3.1.12  setYear(y)

Declaration  Declared in studyplanner.js on line 92:
function setYear(y) {

Description  Sets the beginning year of the user’s study by changing startYear (3.1.7). Also the year numbering in the schedule table is updated.

This function is called when the user changes the beginning year in the Major part.

Parameters

y – The new beginning year (e.g. 2012).

3.1.13  removeYear()

Declaration  Declared in studyplanner.js on line 115:
function removeYear() {


**Description**  Removes the last year of the user’s schedule, and also removes the courses within this year. You cannot get less than three years in your schedule, so if there are only 3 or less years left in the

### 3.1.14 addYear()

**Declaration**  Declared in `studyplanner.js` on line 135:

```javascript
function addYear() {
```

**Description**  Add an empty year to the schedule. You can have at most 10 years in your schedule. When you add the 10th year, the button “Add year” will be disabled, such that you can add no more years.

### 3.1.15 fillInCourses(courses, major)

**Declaration**  Declared in `studyplanner.js` on line 186:

```javascript
function fillInCourses(courses, major) {
```

**Description**  Adds all courses in the `courses` array to the schedule. It is assumed that these courses are from the given major, and the position in the schedule is determined by using this major in combination with its planning. Subjects will only be added when they are not already in the schedule.

This function is used by `fillInSchedule` (3.1.16) to add the major subjects to the schedule table.

**Parameters**

- `courses` – An array with `CourseInfo`’s (3.1.21), that need to be shown in the schedule.
- `major` – The major the courses belong to.

### 3.1.16 fillInSchedule()

**Declaration**  Declared in `studyplanner.js` on line 236:

```javascript
function fillInSchedule() {
```

**Description**  Fill in all major subjects into the schedule. Uses the global variable `major` (3.1.8) to determine the major of a student. If the courses of the user’s major are already known, they are filled in immediately, if not, they are requested from the server.

### 3.1.17 updateMajorErrorBar(bool)

**Declaration**  Declared in `studyplanner.js` on line 294:

```javascript
function updateMajorErrorBar(bool) {
```
CHAPTER 3. JAVASCRIPT

Description  Show error if no major was yet chosen and the user tries to search for deepening/broadening subjects.

Parameters

bool – Whether the “deepening/broadening” button is pressed.

3.1.18  showMajorInput()

Declaration  Declared in studyplanner.js on line 314:
function showMajorInput() {

Description  Show input dialog for choosing a major. Also hides the checkModal.

3.1.19  saveMajor()

Declaration  Declared in studyplanner.js on line 324:
function saveMajor() {

Description  Called when “Save Major” is pressed. Saves the major and the majorName of the user by storing it in global variable major and majorName. Also saves the user’s first year, by updating the years of the table and storing it in global variable startYear.

3.1.20  askMajorCourses()

Declaration  Declared in studyplanner.js on line 345:
function askMajorCourses() {

Description  Request all major courses (of the major indicated by global variable major) from the server, and put them in the majorCourses array. Furthermore the colouring of subjects is updated using colorCourses() (3.13.9).

3.1.21  CourseInfo(code, name, ects, level, lastYear, planning, priorknowledge)

Declaration  Declared in studyplanner.js on line 410:
function CourseInfo(code, name, ects, level, lastYear, planning, priorknowledge) {

Description  Representation of all information belonging to a course.

For example, this could be used as follows:

var c = new Course(c, n, e, level, ly, f, pl, pk)

Then the year of this object, for example, can be accessed using c.year.
Parameters

code – The subject code, like “2IP15.
name – The subject name, like “Programming methodology”.
ecrs – Integer containing the number of ECTS rewarded for this subject.
level – A string with the difficulty level of this course (for example “Basic”.
lastYear – The last year this course will be taught.
planning – An array of tuples when this course is given, where each tuple contains:

  year - The year of the major a normal student would take this course.
  This can be 1, 2 or 3.
target_group - The package or major this subject belongs to (as an
  integer).
packet - The name of the package or major this subject belongs to.
quartile - The quartile the course is given.
timeslot - The timeslot the subject is in (for multiple timeslots, 
  separate them by a “;”).
priorknowledge – A list of course IDs of courses that are considered prior 
  knowledge for this subject.

3.1.22 CoherentPackage(target_group, name)

Declaration  Declared in studyplanner.js on line 429:
function CoherentPackage(target_group, name) {

Description  Representation of a coherent package. This function also contains the variable 
coursesAdded, that indicates whether or not the courses are already requested from the 
server and added to this package.

Parameters

target_group – The ID of the package, as an integer.
name – The name of this package, as a string.

3.1.23 clickedCoherentPackage(event, pack)

Declaration  Declared in studyplanner.js on line 444:
function clickedCoherentPackage(event, pack) {

Description  This function is called when the user clicked on a coherent package. It will 
request and show the subjects inside this coherent package (and request them from the 
server if necessary).
CHAPTER 3. JAVASCRIPT

Parameters

- event – The click event from jQuery.
- pack – The coherent package that was clicked (a jQuery class).

3.1.24 clickedCourse(course)

Declaration  Declared in studyplanner.js on line 469:
function clickedCourse(course) {

Description  This function is called when the user clicked on a course. It will select the
clicked course, and highlight cells in the schedule where this course can be placed. Also
subject information will be shown about the course.

Parameters

- course – The course that was clicked. This is a div element from the GUI.

3.1.25 removeCourse(clickedicon)

Declaration  Declared in studyplanner.js on line 490:
function removeCourse(clickedicon) {

Description  Removes this course from the user’s schedule. This function is called when
the close icon on the course is clicked.

Parameters

- clickedicon – The close button that has been clicked.

3.1.26 clickedCell(event, cell)

Declaration  Declared in studyplanner.js on line 516:
function clickedCell(event, cell) {

Description  This function is when a table cell is clicked. It will move the selected course
to this position. If no course is selected, nothing happens.

Parameters

- event – The click event.
- cell – The cell in the table that was clicked.

3.1.27 setCourses(pack)

Declaration  Declared in studyplanner.js on line 553:
function setCourses(pack) {
3.1. STUDYPLANNER.JS

Description  Request all courses from this package from server.

Parameters

pack – The package from which the courses must be shown; pack must be a jQuery object.

3.1.28  first

Declaration  Declared in studyplanner.js on line 580:
var first = true;

Description  Whether this is the first time a course is clicked, and thus, subject information should be shown.

3.1.29  showSubjectInformation(course)

Declaration  Declared in studyplanner.js on line 592:
function showSubjectInformation(course){

Description  Updates and shows the subject information for the given subject.
  This method handles the subjects that have a subject code starting with “KROK” differently. These subjects do not have a real subject code yet, and are inserted on the server side by the Python method addSoftwareScienceMajor() (6.2.11).

Parameters

course – A course in the schedule (an HTML div, where
course.data(’courseInfo’) must contain subject information.

3.1.30  checkDoubleCourses(cell)

Declaration  Declared in studyplanner.js on line 713:
function checkDoubleCourses(cell) {

Description  Checks if there are 2 courses in this cell that are given in the same timeslot. Courses with timeslot “?” are ignored (they are treated as if they were given in all timeslots).

Parameters

  cell – The cell that needs to be checked (this should be a cell from the HTML table representing the schedule).

3.1.31  warning(str, cell)

Declaration  Declared in studyplanner.js on line 774:
function warning(str, cell) {
CHAPTER 3. JAVASCRIPT

Description Show a warning to the user, depending on the input string.
This method could be extended in the future to allow for different types of warnings,
distinguished by the input string.

Parameters

str – If str = conflict, this function will show a warning about 2 courses in
the same schedule. Else, nothing will happen.
cell – The cell to show the warning on (as a popup).

3.1.32 emptySchedule()

Declaration Declared in studyplanner.js on line 796:
function emptySchedule() {

Description Removes all subjects from the user’s schedule. Also deselects the subject that
is currently selected.

3.1.33 recommend_timeout

Declaration Declared in studyplanner.js on line 822:
var recommend_timeout = null;

Description A variable which is used to postpone recommendation requests, if the user is
currently dragging a subject.

3.1.34 recommendSubjects()

Declaration Declared in studyplanner.js on line 828:
function recommendSubjects() {

Description Ask the server for subjects this user might like, based on his current sched-
ule. Will show these subjects inside the recommend-subjects-container. If the server
recommends no subjects, the entire container is hidden.

3.1.35 setRecommendedSubjects(subjects)

Declaration Declared in studyplanner.js on line 866:
function setRecommendedSubjects(subjects){

Description Sets recommended subjects to subjects.

Parameters

subjects – Array of subjects in standard format, as returned by the server.
3.2 searchSubjects.js

This file contains the functions for searching for subjects and showing the search results to the user.

Variable index  This file contains the following variables:

lastSearchTerm  ........................................ 41

Function index  This file contains the following functions:

getPostDataSearch() ........................................ 41
keywordHighlight(str) ...................................... 43
removeKeywordHighlight() .................................. 42
search() ...................................................... 42
searchButtonWaiting(bool) .................................. 42
showSearchResults(data, results) .......................... 42

Detailed documentation for all declarations in this file follows.

3.2.1 lastSearchTerm

Declaration  Declared in searchSubjects.js on line 9:

```javascript
var lastSearchTerm = '';
```

Description  The last search term the user used to search for subjects or packages. This is used to highlight this word in descriptions and subject names.

3.2.2 getPostDataSearch()

Declaration  Declared in searchSubjects.js on line 27:

```javascript
function getPostDataSearch() {
```

Description  Returns the post data to use for the subject-search AJAX call. This is an internal method used by search().

Returns  postdata, depending on input it contains:

quartile – The selected quartile.
year – The selected year (can be 1, 3 or 3).
major – The user’s major id.
target_group – The user is looking for subjects of this major (this can also be his own major).
searchTerm – The search term entered by the user.
range – Always 0-19, the search results we want to see.

41
broadening – Can be y, n or left out. y means we are looking for broadening subjects, n means deepening, leaving it out means both.
difficulty – The difficulty the user has selected (Basic, Intermediate or Advanced).

3.2.3 search()

Declaration  Declared in searchSubjects.js on line 94:
function search(){

Description  Searches for subjects, using postData to determine the user’s choices and showSearchResults to show the search results.

3.2.4 searchButtonWaiting(bool)

Declaration  Declared in searchSubjects.js on line 121:
function searchButtonWaiting(bool) {

Description  This function will show a throbber in the #search-results and disable the search button to indicate we are now searching, or enable the search button again to show that searching is ready.

Parameters

bool – A boolean. If bool = true a loading animation will be shown and the search button will be disabled. Else, the search button will be enabled again.

3.2.5 showSearchResults(data, results)

Declaration  Declared in searchSubjects.js on line 138:
function showSearchResults(data, results) {

Description  Create subjects from the data and put them in some container. Data should be an array containing search results (so an array of subjects in standard format, see page 160). Then this function will create the given courses and show them in the container results.

Parameters

data – The subjects as returned by the server, which is an array of subjects in standard format.
results – The container to show the results in.

3.2.6 removeKeywordHighlight()
Description  Removes the keyword highlight from all highlighted words on the entire page. This function also removes the <span> around this word.

3.2.7  keywordHighlight(str)

Declaration  Declared in searchSubjects.js on line 178:

```javascript
function keywordHighlight(str){

Description  Highlights the search term of the user (as stored in global variable lastSearchTerm) in str.

Parameters

str – Some string.
CHAPTER 3. JAVASCRIPT

3.3 searchPackages.js

This file contains functions to search for coherent packages and show the search results to
the user.

Function index  This file contains the following functions:

coherentSearchButtonWaiting(bool) ........................................... 44
searchCoherentPackage() ...................................................... 44
showCoherentPackages(data) ............................................... 44

Detailed documentation for all declarations in this file follows.

3.3.1 searchCoherentPackage()

Declaration  Declared in searchPackages.js on line 13:
function searchCoherentPackage(){

Description  Searches for coherent packages. Sends a request to the server containing the
following preferences of the user:

    searchTerm - The user’s search term.
    type - Depending on if the user is looking for USE- or elective packages, this
    variable is 1 or 3.

3.3.2 coherentSearchButtonWaiting(bool)

Declaration  Declared in searchPackages.js on line 47:
function coherentSearchButtonWaiting(bool){

Description  Modifies the coherent-package search button to show that currently searching
is busy, or is ready.

Parameters

    bool – A boolean. If true, a loading animation will be shown and the search
    button will be disabled. Else the search button will be enabled again.

3.3.3 showCoherentPackages(data)

Declaration  Declared in searchPackages.js on line 66:
function showCoherentPackages(data){

Description  Show coherent packages given in data.
3.3. SEARCHPACKAGES.JS

Parameters

data – Data with the coherent packages as returned by the server. Array of dictionaries containing at least the following fields:

   name – The name of this package.
   target_group – The target group number of this package.
CHAPTER 3. JAVASCRIPT

3.4 load.js

This file contains functions that help build the GUI. These are called when the page is loaded.

Function index  This file contains the following functions:

- autoHiding() .................................................. 48
- buildTable(years) ......................................... 46
- doDragging() .................................................. 47
- doDropping() .................................................. 47
- fixTabletMenus() ............................................. 48
- getMajors(fun) ................................................. 47
- load() .......................................................... 46
- receiveMajors(data) ........................................... 47

Detailed documentation for all declarations in this file follows.

3.4.1 load()

Declaration  Declared in load.js on line 18:
function load() {

Description  Called when the document is ready. It will build the table, containing the user’s schedule. By default, it will build a table of 3 years. It will initialize the container showing the subject information and it will try to get the users major courses.

The function also checks if the user is already logged in. If he is, we try loading his schedule from local storage. If this fails, he gets to choose from a list of previously saved schedules (if he has any).

If the user is not logged in we also try getting his schedule from local storage. If this fails, the “Welcome” screen is shown.

3.4.2 buildTable(years)

Declaration  Declared in load.js on line 58:
function buildTable(years) {

Description  Builds the schedule table and puts it in the div with id schedule-container. It removes the previous schedule (and all subjects inside it).

Parameters

- years – The number of years to put in the table.
3.4.3 getMajors(fun)

Declaration  Declared in load.js on line 114:
function getMajors(fun) {

Description  Get a list of all majors from the server. Every major is represented as a string (containing its name) in this list.

Parameters

  fun – Function to be called with the received list of majors.

3.4.4 receiveMajors(data)

Declaration  Declared in load.js on line 134:
function receiveMajors(data) {

Description  Fills the major list (#major) with major names.

Parameters

  data – A list containing majors with name the major’s name and target_group the target group belonging to this major.

3.4.5 doDragging()

Declaration  Declared in load.js on line 164:
function doDragging() {

Description  Makes all subjects draggable. Will drag a clone of the subject, with opacity 0.5. Also sets the start and stop function of the drag, called when the user starts or stops dragging.

  On start, global variable isDragging is set to true and the cells where the subject can be dropped are highlighted.

  On stop, we set isDragging back to false, and the cells are no longer highlighted.

3.4.6 doDropping()

Declaration  Declared in load.js on line 222:
function doDropping() {

Description  Makes all tableInnerCells that are highlighted droppable. Also sets the functions for drop.

  On drop, the dragged course is placed in the cell where it is dropped (also updating which subjects are disabled and recommending new subjects to the user) and * * isDragging is set to false again.
3.4.7 autoHiding()

Declaration  Declared in load.js on line 270:
function autoHiding() {

Description  Sets modal properties of different modals, such that warnings are hidden after
the modal is hidden. This is used to reset the state of the modal after usage.

This is done for the startModal, which is shown when the website is loaded, the
loginModal and the registerModal shown when the user wants to log in or register, and
the saveModal, when the user wants to save a schedule.

3.4.8 fixTabletMenus()

Declaration  Declared in load.js on line 306:
function fixTabletMenus() {

Description  Bootstrap contains a bug that prevents dropdown menus on tablets from being
usable. See https://github.com/twitter/bootstrap/issues/2975 for more information.

This is a workaround for this bug, that should be called on every page load.
This file is responsible for creating the printer-friendly version of the schedule. It will open a new window with this print-version, and open a standard printing dialog. This schedule will be a copy of the original schedule, but with another layout. Also subject codes will be shown in this version (the printversion.css CSS file is responsible for this).

**Variable index**  This file contains the following variables:

- `end` .................................................. 49
- `start` .................................................. 49

**Function index**  This file contains the following functions:

- `getPrintVersion()` ................................. 49

Detailed documentation for all declarations in this file follows.

### 3.5.1 `start`

**Declaration**  Declared in `printversion.js` on line 12:

```javascript
var start = '<!DOCTYPE html>+
```

**Description**  The start string containing the beginning of the HTML of the new page that will show the print version. It contains necessary scripts and the beginning of a div that will contain the schedule.

### 3.5.2 `end`

**Declaration**  Declared in `printversion.js` on line 29:

```javascript
var end = '</div>+'
```

**Description**  End of the HTML on the page showing the printversion. Between start and end you must paste the HTML containing the user’s current schedule.

### 3.5.3 `getPrintVersion()`

**Declaration**  Declared in `printversion.js` on line 37:

```javascript
function getPrintVersion() {
```

**Description**  Creates and opens a printable version of the schedule in schedule-container. This version uses a different CSS file, to make the subject list black-and-white. A print dialog is shown to enable the user to easily print this version.
CHAPTER 3. JAVASCRIPT

3.6 authentication.js

This file contains functions for user management. There are functions for logging a user in or out, and for registering new users.

**Function index**  This file contains the following functions:

- `checkLoggedIn(showStart)` ........................................... 50
- `doLogout()` .................................................................. 50
- `emptyPasswords()` .......................................................... 52
- `isValidPassword(password)` ............................................ 53
- `isValidUsername(username)` ............................................ 52
- `login()` ...................................................................... 51
- `login2()` ..................................................................... 51
- `loginGUI(username)` ....................................................... 51
- `loginToServer(username, password, bool, load)` ............... 51
- `logout()` .................................................................... 50
- `register()` ................................................................... 52

Detailed documentation for all declarations in this file follows.

### 3.6.1 logout()

**Declaration**  Declared in authentication.js on line 9:

```javascript
function logout(){
```

**Description**  Log out a user if he has not made any unsaved modifications, shows a warning otherwise.

### 3.6.2 doLogout()

**Declaration**  Declared in authentication.js on line 22:

```javascript
function doLogout(){
```

**Description**  Log out a user and update the GUI accordingly. Will empty the user’s schedule when he is logged out, and also throw away his local storage.

### 3.6.3 checkLoggedIn(showStart)

**Declaration**  Declared in authentication.js on line 68:

```javascript
function checkLoggedIn(showStart){
```

**Description**  Asks the server if a user is logged in, and if so, set the GUI to show this (using the loginGUI function).
3.6. AUTHENTICATION.JS

Parameters

showStart – Boolean indicating if we should show the startmodal when the user is not yet logged in, load the schedule if the user is logged in and set his major to -1 if something goes wrong. This function is called with true on page load, for other uses it is called with false.

3.6.4 loginGUI(username)

Declaration  Declared in authentication.js on line 135:
function loginGUI(username){

Description  Change the GUI to show that a certain user is logged in. That means hiding the Log in and Register buttons in the navigation bar, and showing the user’s name with a dropdown menu. Furthermore the load and save buttons in the Options menu are no longer disabled.

Parameters

username – The username of the user that is logged in.

3.6.5 login()

Declaration  Declared in authentication.js on line 168:
function login() {

Description  Tries to log in a user at the server. If the server returns that the username and password are correct, the GUI will be changed such that this is reflected (using the loginGUI function), otherwise a warning will be shown.

This function will be called when the Log in button is pressed in the logging-in window.

3.6.6 login2()

Declaration  Declared in authentication.js on line 182:
function login2() {

Description  Same as login (see 3.6.5), but uses other input fields to get username and password.

3.6.7 loginToServer(username, password, bool, load)

Declaration  Declared in authentication.js on line 205:
function loginToServer(username, password, bool, load) {

Description  Tries to log the user in at the server.
CHAPTER 3. JAVASCRIPT

Parameters

username – The username of the user.
password – The password of the user.
bool – boolean Whether we need to show a warning text if the username or password turns out to be incorrect. Otherwise, a warning modal will be shown. This is used as follows: bool = true is given if the user tries to log in by himself, using the log-in modal. bool = false should be given if the application tries to log in the user automatically after he registered. Although this should never fail, in the case that it fails a warning modal will be shown – instead of printing the warning text in the unused log-in modal.
load – Whether the user’s schedule should be loaded after login. load = true should be given if the user logs in immediately after opening the website. If the user logs in himself, load = false should be given.

3.6.8 emptyPasswords()

Declaration  Declared in authentication.js on line 276:

function emptyPasswords() {

Description  Empties the fields that show the user’s password in the log-in modal and the register modal.

3.6.9 register()

Declaration  Declared in authentication.js on line 289:

function register() {

Description  Tries to register a new user at the server.
This function will be called when the Register button is pressed in the register window. It will show a warning if 2 different passwords are given, an invalid username is given (see page 52) or an invalid password is given (see page 53).

3.6.10 isValidUsername(username)

Declaration  Declared in authentication.js on line 360:

function isValidUsername(username) {

Description  Checks whether the given username would be considered valid by the server.

Parameters

username – The username to check.

Returns  Whether the name is valid, meaning it contains between 5 and 20 characters, does not contain two or more dots in a row and only contains 0-9, A-z, _, or -.
3.6.11 isValidPassword(password)

**Declaration**  Declared in authentication.js on line 371:
```javascript
function isValidPassword(password) {
```

**Description**  Checks whether the given password would be considered valid by the server.

**Parameters**

- `password` – The password to check.

**Returns**  Whether the password is valid (contains only ASCII characters and no ASCII control characters).
CHAPTER 3. JAVASCRIPT

3.7 storage.js

This file contains functions to save the schedule to the server, and loading them back. Saving or loading a schedule is only possible if the user is logged in.

Note: the actual login and logout functions are in file authentication.js.

Function index  This file contains the following functions:

- askLoadName(success) ....................................................... 57
- askSaveName(success) ....................................................... 57
- checkSavenameAndSave(savename) ..................................... 56
- getSavedList(success) ..................................................... 55
- loadClick() ................................................................. 54
- loadScheduleFromServer(savename, bool) .......................... 55
- removeClick() .............................................................. 55
- saveAsClick() ............................................................... 54
- saveClick() ................................................................. 54
- saveScheduleToServer(savename) ...................................... 56
- showLoginClick() .......................................................... 55
- showSchedule(data) ....................................................... 55

Detailed documentation for all declarations in this file follows.

3.7.1 loadClick()

Declaration  Declared in storage.js on line 15:
function loadClick() {

Description  This function is called when the user clicks Load. Will then request a list of saved schedules from the server using getSavedList (see page 55). This list will be then shown to the user.

3.7.2 saveClick()

Declaration  Declared in storage.js on line 55:
function saveClick() {

Description  This function is called when the user clicks Save. If the name of the schedule is not known, behaves the same as saveAsClick (see page 54), otherwise saves the schedule under the last known name.

3.7.3 saveAsClick()

Declaration  Declared in storage.js on line 67:
function saveAsClick() {

54
Description  This function is called when the user clicks `Save as`. Asks a name for the schedule, so it can be saved.

### 3.7.4 showLoginClick()

**Declaration**  Declared in `storage.js` on line 79:

```javascript
function showLoginClick() {

```

Description  This function is called when the user clicks the `log in` link in a warning message. It hides all alerts and shows the login modal.

### 3.7.5 getSavedList(success)

**Declaration**  Declared in `storage.js` on line 94:

```javascript
function getSavedList(success) {

```

Description  Fetches a list of saved schedules from the server. Since this is an asynchronous operation, a callback function should be given that is executed on success. When there is an error, the callback is not executed and instead a warning message will be shown.

**Parameters**

- `success` – Callback to be executed when the schedule list is fetched.

**Returns**  An array containing names of saved schedules.

### 3.7.6 removeClick()

**Declaration**  Declared in `storage.js` on line 117:

```javascript
function removeClick() {

```

Description  Remove the schedule selected.

### 3.7.7 loadScheduleFromServer(savename, bool)

**Declaration**  Declared in `storage.js` on line 150:

```javascript
function loadScheduleFromServer(savename, bool) {

```

Description  Request a schedule from the server, from the user that is currently logged in. If this fails, the function will check if the user is actually still logged and log him out if not.
CHAPTER 3. JAVASCRIPT

Parameters

savename – A string containing the name of the schedule to load.
bool – A boolean. If bool == false, no warning will be shown on failure (or when the schedule does not exist), and the major will then be reset to -1. If however bool == true a warning will be shown on failure and everything else will remain unchanged.

3.7.8 checkSavenameAndSave(savename)

Declaration   Declared in storage.js on line 227:
function checkSavenameAndSave(savename) {

Description   Checks whether the save name already exists. If not, the schedule is saved immediately. Else, a dialog is shown asking whether the schedule should be saved, and if the user clicks Yes, the schedule is saved as well.

Parameters

savename – A string containing the name to save the schedule with on the server.

3.7.9 saveScheduleToServer(savename)

Declaration   Declared in storage.js on line 269:
function saveScheduleToServer(savename) {

Description   Saves the current schedule to the server with the given name.

Parameters

savename – A string containing the name to save the schedule with on the server.
if saving fails, the function will check if the user is still logged in and show a warning.

3.7.10 showSchedule(data)

Declaration   Declared in storage.js on line 310:
function showSchedule(data) {

Description   Puts the given schedule actually in the schedule table, and handles everything about that (removing the error popovers, colouring subjects and so on).

Parameters

data – An array data with arrays $s_i$ of subjects in standard format. Where $s_i$ contains the subjects given in quartile $i$, where quartile $i$ means $\text{year} = \lfloor \frac{i}{4} \rfloor + 1$ and quartile is $i + 1 - 4 \ast (\text{year} - 1)$.
3.7.11 askSaveName(success)

Declaration  Declared in storage.js on line 388:
function askSaveName(success) {

Description  Shows a dialog asking for a name to save the schedule with. Will only accept save names that are not empty.

Parameters

success – Callback to execute when the name has been entered. When the dialog is dismissed instead, the callback will not be called.

3.7.12 askLoadName(success)

Declaration  Declared in storage.js on line 411:
function askLoadName(success) {

Description  Shows a dialog asking for the name of the schedule to load.

Parameters

success – Callback to execute when the name has been entered. When the dialog is dismissed instead, the callback will not be called.
3.8 persistentData.js

This file contains functions to maintain some properties from the current state when refreshing the page. This means that the current (possibly unsaved) schedule can be recovered when the page is closed and then opened again. However, the state will only be recovered if the login state is the same when recovering the state as when the state was saved. We also apply an expiration time on the state.

Note that recovering the previous state must be done before the current state has been written to the storage. → updatePersistentData → loadPersistentData

The current state is automatically written when setModified is called, so no feature should have to call a function from this file for that purpose.

A state in this file is defined as follows.

- schedule: The schedule as a list of lists of subjects (grouped by quartile) in minimal form. The representation is consistent with getSchedule;
- major, majorName: The major and its name: major, majorName;
- year: The first year in the schedule table: startYear;
- schedName: The name of the current schedule (if this exists, otherwise this will be the empty string): currentScheduleName;
- modSchedule: A boolean representing if the schedule was modified: modifiedSchedule;
- loggedIn: A boolean indicating if the user was logged in: isLoggedIn. This is used to only recover the previous state if the login state is the same;

Variable index This file contains the following variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Located on line</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXPIRE_SECONDS</td>
<td>58</td>
</tr>
<tr>
<td>isPersistentSupported</td>
<td>60</td>
</tr>
</tbody>
</table>

Function index This file contains the following functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Located on line</th>
</tr>
</thead>
<tbody>
<tr>
<td>emptyPersistentData()</td>
<td>59</td>
</tr>
<tr>
<td>getPersistentData()</td>
<td>60</td>
</tr>
<tr>
<td>isPersistentDataSupported()</td>
<td>60</td>
</tr>
<tr>
<td>loadPersistentData()</td>
<td>59</td>
</tr>
<tr>
<td>updatePersistentData()</td>
<td>59</td>
</tr>
</tbody>
</table>

Detailed documentation for all declarations in this file follows.

3.8.1 EXPIRE_SECONDS

Declaration Declared in persistentData.js on line 30:

EXPIRE_SECONDS = 60*60;
3.8. PERSISTENTDATA.JS

Description  The time in seconds before a state will expire and cannot be recovered.

3.8.2  updatePersistentData()

Declaration  Declared in persistentData.js on line 46:
function updatePersistentData() {

Description  A function that will write the current state to the “local storage” (an HTML 5
feature supported by most browsers, see http://www.w3.org/TR/webstorage/#the-localstorage-
attribute). The localStorage attribute does not support expiration and therefore, we also
write the current date in seconds from the UNIX epoch to the local storage.

This function does nothing when localStorage is not supported. → isPersistentDataSupported()

We use the keys persistent_data and expiry_date from the localStorage attribute.
This function is called by setModified, and therefore by almost every function that changes
something in the schedule. Thus one should be careful not to call updatePersistentData
before loadPersistentData. → setModified

3.8.3  loadPersistentData()

Declaration  Declared in persistentData.js on line 79:
function loadPersistentData() {

Description  A function to recover the previous state, if possible, and will return a boolean
value indicating if the function could recover the previous state. This function will initialize
the major and the start year, even if the previous state could not be recovered.

This function can be inhibited to recover the previous state if the current login state was
not the same as the previous login state, or no previous state is present (first visit, the state
has been expired or the user has logged out and closed the website in the previous session).
→ emptyPersistentData No state can also be recovered when localStorage is not supported
on the browser. → isPersistentDataSupported()

As the schedule saved by updatePersistentData is in minimal form, the function
enqueues an AJAX request to url_converter, which will convert the minimal form to
standard form, before the schedule is recovered. Therefore this function is asynchronous.

Returns  A boolean that will be true if the state is recovered and an AJAX request is
enqueued to url_converter (which can still fail). Otherwise the value will be false.

3.8.4  emptyPersistentData()

Declaration  Declared in persistentData.js on line 134:
function emptyPersistentData() {
CHAPTER 3. JAVASCRIPT

**Description**  Empties the current state. Caution should be used when calling this function, as any change to the schedule (setMajor, setModified, and so on) will update the saved state by calling updatePersistentData. Does nothing when localStorage is not supported. → isPersistentDataSupported()

### 3.8.5 getPersistentData()

**Declaration**  Declared in persistentData.js on line 152:

```javascript
function getPersistentData() {

Description  Will return a JavaScript object encoded in a JSON string, representing the saved state. This string can be null or ‘null’ if the state was not saved.

The storage will return null if the save date corresponding to the saved state was more than EXPIRE SECONDS seconds ago. The JavaScript object returned will have the fields as defined in the file description of persistentData.js.

Returns  A JavaScript object encoded as JSON string with the previous state, if this could be recovered.

Returns null when localStorage is not supported. → isPersistentDataSupported()

### 3.8.6 isPersistentSupported

**Declaration**  Declared in persistentData.js on line 171:

```javascript
isPersistentSupported = null;
```

Description  A cache used by isPersistentDataSupported() to cache its results.

### 3.8.7 isPersistentDataSupported()

**Declaration**  Declared in persistentData.js on line 178:

```javascript
function isPersistentDataSupported() {

Description  Check whether HTML 5 local storage is supported on this web browser.

Returns  A boolean determining if the local storage is supported.
3.9 validator.js

This file contains functionality to show if the user’s current schedule is valid. Checking if the schedule is valid is mostly done by the server, only the number of ECTS is checked on the client-side of our application. So this file mainly contains functions to show in the GUI whether a schedule is valid, depending on input data that is requested from the server.

**Function index**  This file contains the following functions:

- `checkCoherent(data)`  
- `checkEcts(courses)`  
- `checkMajor(data)`  
- `checkOverlap(data)`  
- `checkUse(data)`  
- `countEcts(courses)`  
- `resetTableRows()`  
- `validateSchedule()`  

Detailed documentation for all declarations in this file follows.

### 3.9.1 validateSchedule()

**Declaration**  Declared in `validator.js` on line 21:

```javascript
function validateSchedule() {
```

**Description**  Called when the user clicks Validate schedule. It will check the number of ECTS of the schedule, and if the major of the user is known, it will try to get the following information from the server:

- If the user has enough coherent packages and which one.
- If the user has enough USE packages (and which one).
- If the user has all major courses in his schedule.
- If there are subjects that are used in multiple (coherent/USE) packages (this might not be approved by the examination committee).

### 3.9.2 checkOverlap(data)

**Declaration**  Declared in `validator.js` on line 65:

```javascript
function checkOverlap(data) {
```

**Description**  Shows if there are subjects that belong to more than one (Elective/USE) package.
Parameters

data – Data about the overlapping courses, containing a status which can be PASS or FAIL, and overlap is a list of subject codes that overlap.

3.9.3 checkMajor(data)

Declaration  Declared in validator.js on line 85:

```
function checkMajor(data) {
```

Description  Shows the user if all major courses are in his schedule.

Parameters

data – Data about the major courses, containing status which can be PASS or FAIL and missing which is an array of subject codes.

3.9.4 checkUse(data)

Declaration  Declared in validator.js on line 105:

```
function checkUse(data) {
```

Description  Checks if the user has enough USE packages and shows this in the GUI.

Parameters

data – Data about USE packages in the schedule as returned by the server. Must contain status, which is PASS or FAIL and packages with a list of USE packages in the user’s schedule.

3.9.5 checkCoherent(data)

Declaration  Declared in validator.js on line 127:

```
function checkCoherent(data) {
```

Description  Checks if the user has enough coherent packages and shows this in the GUI.

Parameters

data – Data about coherent packages in the schedule as returned by the server. Must contain status, which is PASS or FAIL and packages with a list of coherent packages in the user’s schedule.

3.9.6 resetTableRows()

Declaration  Declared in validator.js on line 153:

```
function resetTableRows() {
```
3.9. VALIDATOR.JS

Description  Reset all table rows by putting throbbers in it. That means, remove any previous validation results.

3.9.7  countEcts(courses)

Declaration  Declared in validator.js on line 171:
function countEcts(courses) {

Description  Counts the total number of ECTS in the schedule.

Parameters

  courses – An array of information of courses in the schedule (CourseInfo).

Returns  An integer indicating the ECTS count.

3.9.8  checkEcts(courses)

Declaration  Declared in validator.js on line 183:
function checkEcts(courses) {

Description  Show the user if there are enough ECTS in his schedule.

Parameters

  courses – Array of CourseInfo’s (see page 36) currently in the schedule.
CHAPTER 3. JAVASCRIPT

3.10 urls.js

This file contains the URLs to post to on the server side.
   Every operation has its own variable, that indicates the URL for this operation.
   Compare this to the Python script KroketApp.query.urls, that does approximately the
   same thing on the server side.

Variable index  This file contains the following variables:

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Declared in urls.js on line</th>
</tr>
</thead>
<tbody>
<tr>
<td>url_converter</td>
<td>12</td>
</tr>
<tr>
<td>url_elPackageList</td>
<td>17</td>
</tr>
<tr>
<td>url_isLoggedIn</td>
<td>12</td>
</tr>
<tr>
<td>url_load</td>
<td>12</td>
</tr>
<tr>
<td>url_login</td>
<td>12</td>
</tr>
<tr>
<td>url_logout</td>
<td>12</td>
</tr>
<tr>
<td>url_majorList</td>
<td>12</td>
</tr>
<tr>
<td>url_majorSubjects</td>
<td>12</td>
</tr>
<tr>
<td>url_packageSearch</td>
<td>12</td>
</tr>
<tr>
<td>url_packageSubjects</td>
<td>12</td>
</tr>
<tr>
<td>url_recommend</td>
<td>12</td>
</tr>
<tr>
<td>url_register</td>
<td>12</td>
</tr>
<tr>
<td>url_removeSchedule</td>
<td>12</td>
</tr>
<tr>
<td>url_save</td>
<td>12</td>
</tr>
<tr>
<td>url_savedList</td>
<td>12</td>
</tr>
<tr>
<td>url_subjectInfo</td>
<td>12</td>
</tr>
<tr>
<td>url_subjectSearch</td>
<td>12</td>
</tr>
<tr>
<td>url_usePackageList</td>
<td>12</td>
</tr>
<tr>
<td>url_validate</td>
<td>12</td>
</tr>
</tbody>
</table>

Detailed documentation for all declarations in this file follows.

3.10.1 url_majorList

Declaration  Declared in urls.js on line 12:
var url_majorList = '/studieplanner/query/major/list';

Description  A string containing an URL. Posting to this URL gives a list of all major names.

3.10.2 url_elPackageList

Declaration  Declared in urls.js on line 17:
var url_elPackageList = '/studieplanner/query/electivepackage/list';

Description  A string containing an URL. Posting to this URL gives a list of all elective packages and the subjects in the package.
3.10.3 url_usePackageList

Declaration  Declared in urls.js on line 22:
var url_usePackageList = '/studieplanner/query/usepackage/list';

Description  A string containing an URL. Posting to this URL gives a list of all use packages and the subjects in the package.

3.10.4 url_majorSubjects

Declaration  Declared in urls.js on line 27:
var url_majorSubjects = '/studieplanner/query/major/subjects';

Description  A string containing an URL. Posting to this URL gives a list of all mandatory subjects in a given major.

3.10.5 url_packageSubjects

Declaration  Declared in urls.js on line 32:
var url_packageSubjects = '/studieplanner/query/package/subjects';

Description  A string containing an URL. Posting to this URL gives a list of all subjects in a given package.

3.10.6 url_subjectSearch

Declaration  Declared in urls.js on line 37:
var url_subjectSearch = '/studieplanner/query/subject/search';

Description  A string containing an URL. In this way you can search for subjects.

3.10.7 url_packageSearch

Declaration  Declared in urls.js on line 42:
var url_packageSearch = '/studieplanner/query/package/search';

Description  A string containing an URL. In this way you can search for packages.

3.10.8 url_subjectInfo

Declaration  Declared in urls.js on line 47:
var url_subjectInfo = '/studieplanner/query/subject/info';

Description  A string containing an URL. Posting to this URL gives subject information about a subject with given subject code.
CHAPTER 3. JAVASCRIPT

3.10.9 url_load

Declaration  Declared in urls.js on line 52:
var url_load = '/studieplanner/query/schedule/load';

Description  A string containing an URL. Used to save the schedule of a user.

3.10.10 url_save

Declaration  Declared in urls.js on line 57:
var url_save = '/studieplanner/query/schedule/save';

Description  A string containing an URL. Used to load the schedule of a user.

3.10.11 url_savedList

Declaration  Declared in urls.js on line 62:
var url_savedList = '/studieplanner/query/schedule/list';

Description  A string containing an URL. Used to get a list of saved schedules of a user.

3.10.12 url_recommend

Declaration  Declared in urls.js on line 67:
var url_recommend = '/studieplanner/query/recommendation/subject';

Description  A string containing an URL. Used to get recommended subjects from the server.

3.10.13 url_removeSchedule

Declaration  Declared in urls.js on line 72:
var url_removeSchedule = '/studieplanner/query/schedule/delete';

Description  A string containing an URL. Removes a schedule from the server.

3.10.14 url_isLoggedIn

Declaration  Declared in urls.js on line 77:
var url_isLoggedIn = '/studieplanner/query/user/info';

Description  A string containing an URL. Returns if a user is logged in and also what his username is.
3.10 URLs.js

3.10.15 url_login

Declaration  Declared in urls.js on line 82:
var url_login = '/studieplanner/query/user/login';

Description  A string containing an URL. Used to log in a user. Needs a username and password and will return whether these are correct.

3.10.16 url_register

Declaration  Declared in urls.js on line 87:
var url_register = '/studieplanner/query/user/register';

Description  A string containing an URL. Used to register a user. Needs a username and password and will return whether the username is unique.

3.10.17 url_logout

Declaration  Declared in urls.js on line 92:
var url_logout = '/studieplanner/query/user/logout';

Description  A string containing an URL. Used to log out a logged in user. Needs no information from the client.

3.10.18 url_validate

Declaration  Declared in urls.js on line 97:
var url_validate = '/studieplanner/query/schedule/validate';

Description  A string containing an URL. Used to ask if a users schedule is valid.

3.10.19 url_converter

Declaration  Declared in urls.js on line 103:
var url_converter = '/studieplanner/query/subject/convert';

Description  A string containing an URL. Used to convert a list of subjects in minimal format to a list of standard format.
CHAPTER 3. JAVASCRIPT

3.11 translation.js

The English translation of the JavaScript strings.

This file contains a single variable, called _ (a single underscore), that is functioning like a dictionary in which the strings can be found. For example,

   _year  returns “Year”

if this file has been loaded. However if another translation file is loaded, for example translation-nl_NL.js, it will return the translation in the appropriate language.

Some keys, such as aScheduleOfNameXDoesntExist, are actually functions, because these strings need ‘parameters’ (for example the name of the schedule). These are then filled in in the return value. For example,

   _.aScheduleOfNameXDoesntExist('blah')
   returns “A schedule of name blah doesn’t exist.”.

In this way, the HTML file for a certain language is able to include just one of the translation JavaScript files, and the correct translation will be included for the JavaScript strings throughout the application.

Variable index  This file contains the following variables:

   _  ................................................................. 69

Detailed documentation for all declarations in this file follows.

3.11.1 _

Declaration  Declared in translation.js on line 31:

   var _ = {

Description  The object containing the translations.
3.12 translation-nl_NL.js

The Dutch translation of the JavaScript strings. See the documentation for translation.js for more details.

**Variable index** This file contains the following variables:

```javascript
var _ = {
```

Detailed documentation for all declarations in this file follows.

### 3.12.1 `_`

**Declaration** Declared in translation-nl_NL.js on line 9:

```javascript
var _ = {
```

**Description** The object containing the translations.
CHAPTER 3. JAVASCRIPT

3.13 utils.js

This file contains some utility functions.

Variable index This file contains the following variables:

hiddenTab ................................................... 73

Function index This file contains the following functions:

addDeleteIcon(course) ........................................ 75
checkDoubleCoursesRecursive(courses, index) ............ 76
colorCourses() ................................................. 72
disableCourses() ............................................. 73
filterPackageName(name) ...................................... 73
filterTimeSlot(slot) .......................................... 73
getCloneslot(course) ......................................... 78
getSchedule(onlyElective) ..................................... 74
hideDisabler() ................................................. 71
hideStartModal() .............................................. 72
highlight(course) ............................................ 75
makePackage(data) ............................................ 75
makeSubject(data) ............................................. 77
noConflict(courses) .......................................... 77
noCurrConflict(c1, c2) ...................................... 78
packageTypeToString(number) ................................ 70
removeHighlight() .......................................... 75
setExpanded(expanded) ....................................... 73
setMajor(m, mName) ........................................... 71
setModified(bool) ............................................ 71
setSubjectInfo(info) ......................................... 74
showAlert(str) ............................................... 72
showDisabler(text) .......................................... 71
showRegisterModal() ......................................... 72
subjectInfoPopup() .......................................... 75

Detailed documentation for all declarations in this file follows.

3.13.1 packageTypeToString(number)

Declaration Declared in utils.js on line 14:
function packageTypeToString(number) {

70
Description  Converts a number (1, 2, 3) from the server to a readable package type. For example, `packageTypeToString(1)` returns “USE package”.

The server uses these numbers internally to avoid saving the string every time. This function should be called on all places where such a number is shown to the user.

Parameters

number – The number to convert.

### 3.13.2 setMajor(m, mName)

Declaration  Declared in `utils.js` on line 33:
```javascript
function setMajor(m, mName) {
```

Description  This function is used to set the user’s current major to `mName` with target group `m`. If the major is known (meaning `m` is not `-1`) the subjects belonging to the user’s major are requested from the server using `askMajorCourses`, see 3.1.20. The function will also show or hide warnings, depending on the given major.

Parameters

- `m` – Integer representing the major. `m == -1` means the major is unknown.
- `mName` – The name of the major, as a string.

### 3.13.3 showDisabler(text)

Declaration  Declared in `utils.js` on line 66:
```javascript
function showDisabler(text) {
```

Description  Shows the “disabler pane” on top of the schedule.

Parameters

- `text` – The text to show in the disabler pane.

### 3.13.4 hideDisabler()

Declaration  Declared in `utils.js` on line 75:
```javascript
function hideDisabler() {
```

Description  Hides the “disabler pane”.

### 3.13.5 setModified(bool)

Declaration  Declared in `utils.js` on line 91:
```javascript
function setModified(bool) {
```
CHAPTER 3. JAVASCRIPT

Description  Set modifiedSchedule to bool. If bool = true the schedule has been modified since the last time the user saved the schedule, so a warning will be shown when the user tries to leave the webpage or close the browser. Otherwise, this will not happen.

This function is also used to send signals to updatePersistentData, see 3.8.2, which might change the state of some variables depending on the modifications made. Therefore this function should be called AFTER the schedule has been modified.

Parameters

  bool – Whether the ‘modified’ value should be put on true or false.

3.13.6  showAlert(str)

Declaration  Declared in utils.js on line 108:
  function showAlert(str) {

Description  Shows an alert with the given string.

Parameters

  str – HTML string that needs to be shown.

3.13.7  showRegisterModal()

Declaration  Declared in utils.js on line 116:
  function showRegisterModal() {

Description  Shows the register modal.

3.13.8  hideStartModal()

Declaration  Declared in utils.js on line 124:
  function hideStartModal() {

Description  Hides the start modal.

3.13.9  colorCourses()

Declaration  Declared in utils.js on line 132:
  function colorCourses() {

Description  Adds class .major to all subjects that are currently stored in majorCourses (if they are known). Subjects are compared using their subject code, so we do assume this is unique.
3.13. UTILS.JS

### 3.13.10 disableCourses()

**Declaration**  Declared in `utils.js` on line 159:

```javascript
function disableCourses() {
```

**Description**  Searches through the recommended subjects and the subjects in the search results, and disables all subjects that are currently in the schedule. All subjects that are not in the schedule will be enabled.

### 3.13.11 hiddenTab

**Declaration**  Declared in `utils.js` on line 197:

```javascript
var hiddenTab = null;
```

**Description**  The variable stores the tab of the Subject information window that was selected the previous time this information was hidden. When the Subject information expands, this is the tab that will be selected.

  Used by setExpanded, see 3.13.12.

### 3.13.12 setExpanded(expanded)

**Declaration**  Declared in `utils.js` on line 204:

```javascript
function setExpanded(expanded) {
```

**Description**  Toggles the visibility of the Subject information window.

**Parameters**

- `expanded` – If `true`, the window will be expanded; if `false`, it will be hidden.

### 3.13.13 filterPackageName(name)

**Declaration**  Declared in `utils.js` on line 245:

```javascript
function filterPackageName(name) {
```

**Description**  This function removes a “(Bachelor College)” prefix from a package name.

**Deprecated:**  *This is not necessary anymore, since the server already handles this.*

**Parameters**

- `name` – Name of a package or major.

### 3.13.14 filterTimeSlot(slot)

**Declaration**  Declared in `utils.js` on line 256:

```javascript
function filterTimeSlot(slot) {
```
CHAPTER 3. JAVASCRIPT

Description  This function formats the timeslot string from the server. The function replaces all “;” in the slot by “ and “.

Parameters

    slot – The timeslot as a string.

Returns  The formatted timeslot, the format depends on the language of the page, in English all “;” characters have been replaced by “ and “.

3.13.15  getSchedule(onlyElective)

Declaration  Declared in utils.js on line 271:
function getSchedule(onlyElective){

Description  Returns which subjects are currently in the schedule

Returns  An array of length

4 · (number of years in schedule)

containing arrays of subject codes within every quartile in this schedule. For example a three-year-schedule could result in ```[['vak1','vak2'] ['vak3']] [['vak4']]```.

Parameters

    onlyElective – boolean, if true, only non-major courses will be returned.
    (courses that do not have css-class .major).

3.13.16  setSubjectInfo(info)

Declaration  Declared in utils.js on line 308:
function setSubjectInfo(info){

Description  Adds the specified information to the Subject information window.

Parameters

    info – A dictionary containing information about the course. It should contain
    the following fields:

        content – the content description of the subject.
        studyGoal – the study goal description of the subject.
        evaluation – the evaluation of this subject, either ’unknown’ or a
        number from 1 to 10.
3.13. UTILS.JS

3.13.17 subjectInfoPopup()

Declaration  Declared in utils.js on line 336:
function subjectInfoPopup() {

Description  Manages the popup behaviour of the Subject information window. This function
should be called when the user clicks a subject.
   If this is the first time a subject is clicked, the window will appear (using the setExpanded
function). If the user, on the other hand, has already clicked the window to hide it, it will
not appear again.

3.13.18 highlight(course)

Declaration  Declared in utils.js on line 359:
function highlight(course) {

Description  Highlights the places in the table where the given subject can be placed. Also
shows the “Click or drag here to place” placeholder text.

Parameters

   course – The course that is selected by the user.

3.13.19 removeHighlight()

Declaration  Declared in utils.js on line 387:
function removeHighlight() {

Description  Removes all highlights from the cells in the table (and hides the text “Click
to place subject”). Hiding the text does not work in browsers that do not support the CSS
opacity property (but all modern browsers do).
   So after running this function, there are no objects left that have class .highlight.

3.13.20 addDeleteIcon(course)

Declaration  Declared in utils.js on line 399:
function addDeleteIcon(course) {

Description  Adds a delete icon to the given subject and makes sure the function removeCourse
(see 3.1.25) is called when the user clicks this icon.

Parameters

   course – The course to which the close icon should be added.
CHAPTER 3. JAVASCRIPT

3.13.21 checkDoubleCoursesRecursive(courses, index)

Declaration  Declared in utils.js on line 430:
function checkDoubleCoursesRecursive(courses, index){

Description  Recursive function to check for double courses. The courses array contains information about the courses.

Parameters

courses – A list of arrays containing (at least):
  
  code – The subject code.
  slots – An array of the available timeslots (see below).
  currSlot – The first available timeslot.

If a course could be taken in multiple timeslots, but you need only one of them to take the course, these timeslots have to be put in the slots array separately. If, on the other hand, one course instance covers multiple timeslots (for example if it is 16 hours per week), they should form one element of the slots array, separated by semicolons. An element of slots may look like ‘A;B’, for example. Summarizing, the slots array could look like

  ['A', 'A;B', 'C;D;E', 'C'].

index – Should be initially called with 0. Index of the first course in courses that has not yet been planned.

Returns  If there is a setting such that no 2 courses are in the same timeslot.

3.13.22 makePackage(data)

Declaration  Declared in utils.js on line 458:
function makePackage(data){

Description  Create a coherent package (which is a div element that can be placed on the web page, containing data).

Parameters

data – A JSON object with the following keys:

  name – The name of this package.
  target_group – The target group number of this package.

Returns  A div with the layout of a package. The div also has data with key “packageInfo” which contains the class CoherentPackage (see 3.1.22), with the name and target_group from the given data.
### 3.13.23 makeSubject(data)

**Declaration**  Declared in **utils.js** on line 489:

```javascript
function makeSubject(data) {
```

**Description**  Create a subject (div element that can be placed on the web page) with jQuery data.

**Parameters**

- `data` – A JSON object with the following keys (order is irrelevant, keys that are not listed below are ignored):
  - `name` – The name of this subject.
  - `code` – The subject code.
  - `knowledge` – (Optional) An array of subject codes from subjects you need to take this course.
  - `priorKnowledge` – (Optional) A string of subject codes from subjects you need to take this course, seperated by “;”.
  - `planning` – An array of JSON objects containing `year`, `quartile`, `timeSlot`, `target_group` (major or package number) and `packet` (the major or package name).
  - `ects` – The number of ECTS of this course.
  - `difficulty` – Difficulty level of a course.
  - `lastYear` – Last year a course is given (string in the format 'yyyy', or an integer).

**Returns**  A div with the layout of a package. The div also has as `courseInfo` the class `CourseInfo`, with the information from the given data.

### 3.13.24 noConflict(courses)

**Declaration**  Declared in **utils.js** on line 544:

```javascript
function noConflict(courses){
```

**Description**  Checks if all timeslots in `currSlot` are different, so there are currently no conflicting courses. This method supports cases where subjects cover multiple timeslots. It uses `noCurrConflict`, that doesn’t cover these cases.

**Parameters**

- `courses` – A list of arrays containing (at least):
  - `code` – The subject code
  - `currSlot` – One of the available timeslots
CHAPTER 3. JAVASCRIPT

Returns A boolean containing whether there is a conflict.

3.13.25 noCurrConflict(c1, c2)

Declaration Declared in utils.js on line 564:
function noCurrConflict(c1, c2){

Description Returns if arrays c1 and c2 are completely different (containing no double elements).

Parameters

  c1 – An array, probably containing subject codes.
  c2 – Another array.

Returns A boolean whether c1 and c2 are different (there is no element in c1 that is also in c2 and vica versa).

3.13.26 getClone(course)

Declaration Declared in utils.js on line 585:
function getClone(course){

Description Generates a clone of the given course. Will set the onClick to our click function as it is now, not to the function of the given course! Copy’s the courseInfo of the given course

Parameters

  course – The course to be cloned.

Returns A subject with the same css-properties, the same looks, and the same data with key courseInfo as course. Also adds an onclick function to the subject, which calls the function clickedCourse.
Chapter 4

Python (Django) documentation

In this chapter, all of the Python code of the Django internals on the server side will be documented.

Python declarations are ordered on the file they are in.

4.1 KroketApp.models

A Python script which defines the database structure. In this script all models regarding data fetched from the OWIS/onderwijs.tue.nl/ownfo.tue.nl server are defined. This is also the Python script which the Django-server imports to find all the defined models. As such, this script imports all other model-scripts. → KroketApp.savemodels, → KroketApp.recommodels → KroketApp.schedule.daily_jobs.subject_updater.addData

The models defined in this Python script are either entities, weak-entities or relations. As a reminder, entities are models which can be identified using their own (non-foreign) fields; weak-entities can only be a combination of their own fields and foreign-fields; Relations are primarily built of foreign keys and create relations between (weak-)entities.

In the current state the models only support one language (English). In the future some fields might be changed into one-to-many relations, to support multiple languages. (A field \( f \) from model \( m \) will then be a weak-entity with field \( f \), a foreign key to \( m \) and a language identifier). But again, this is not yet supported.

As Django does not support primary-key-sets, but always needs a one-field primary key, this would imply that all our models would be entities, and no relations or weak-entities are allowed to exist. To overcome this, we abstract from certain ‘id’ fields and define in the comments what the primary key-set is and enforce this by adding ‘unique’ constraints.

Hierarchical member index  This file contains the following members:

<table>
<thead>
<tr>
<th>Class</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoursePlanning</td>
<td>93</td>
</tr>
<tr>
<td>id</td>
<td>93</td>
</tr>
<tr>
<td>subject</td>
<td>93</td>
</tr>
<tr>
<td>target_group</td>
<td>93</td>
</tr>
<tr>
<td>year</td>
<td>94</td>
</tr>
</tbody>
</table>
CHAPTER 4. PYTHON (DJANGO)

<table>
<thead>
<tr>
<th>Field</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>quartile</td>
<td>94</td>
</tr>
<tr>
<td>timeslot</td>
<td>94</td>
</tr>
<tr>
<td>mandatory</td>
<td>94</td>
</tr>
<tr>
<td><strong>unicode</strong></td>
<td>94</td>
</tr>
<tr>
<td>getPlanning</td>
<td>95</td>
</tr>
<tr>
<td>Meta</td>
<td>95</td>
</tr>
<tr>
<td>Keyword</td>
<td>88</td>
</tr>
<tr>
<td>name</td>
<td>88</td>
</tr>
<tr>
<td>total_count</td>
<td>88</td>
</tr>
<tr>
<td><strong>unicode</strong></td>
<td>88</td>
</tr>
<tr>
<td>KeywordPacketRelation</td>
<td>100</td>
</tr>
<tr>
<td>id</td>
<td>101</td>
</tr>
<tr>
<td>keyword</td>
<td>101</td>
</tr>
<tr>
<td>target_group</td>
<td>101</td>
</tr>
<tr>
<td>count</td>
<td>101</td>
</tr>
<tr>
<td><strong>unicode</strong></td>
<td>101</td>
</tr>
<tr>
<td>Meta</td>
<td>102</td>
</tr>
<tr>
<td>KeywordSubjectRelation</td>
<td>98</td>
</tr>
<tr>
<td>id</td>
<td>99</td>
</tr>
<tr>
<td>keyword</td>
<td>99</td>
</tr>
<tr>
<td>subject</td>
<td>99</td>
</tr>
<tr>
<td>count</td>
<td>99</td>
</tr>
<tr>
<td><strong>unicode</strong></td>
<td>100</td>
</tr>
<tr>
<td>Meta</td>
<td>100</td>
</tr>
<tr>
<td>Subject</td>
<td>81</td>
</tr>
<tr>
<td>code</td>
<td>82</td>
</tr>
<tr>
<td>name</td>
<td>82</td>
</tr>
<tr>
<td>niveau</td>
<td>82</td>
</tr>
<tr>
<td>ects</td>
<td>82</td>
</tr>
<tr>
<td>remarks</td>
<td>83</td>
</tr>
<tr>
<td>study_goal</td>
<td>83</td>
</tr>
<tr>
<td>content</td>
<td>83</td>
</tr>
<tr>
<td>weekly_content</td>
<td>83</td>
</tr>
<tr>
<td>last_year</td>
<td>83</td>
</tr>
<tr>
<td>video_mat_descr</td>
<td>84</td>
</tr>
<tr>
<td>video_url</td>
<td>84</td>
</tr>
<tr>
<td>url</td>
<td>85</td>
</tr>
<tr>
<td>education_type</td>
<td>85</td>
</tr>
<tr>
<td>examination_type</td>
<td>85</td>
</tr>
<tr>
<td>department</td>
<td>85</td>
</tr>
<tr>
<td>subdepartment</td>
<td>86</td>
</tr>
<tr>
<td>rating</td>
<td>86</td>
</tr>
<tr>
<td>NON_STANDARD_FORMAT_FIELDS</td>
<td>86</td>
</tr>
<tr>
<td>objects</td>
<td>86</td>
</tr>
</tbody>
</table>

80
4.1. KROKETAPP.MODELS

getFollowUp ........................................... 87
getPriorKnowledge ................................. 87
__unicode__ ........................................ 87
SubjectFollowUpRelation ...................... 95
  id ................................................. 96
  subject ......................................... 96
  follow_up ........................................ 96
  __unicode__ .................................... 96
  Meta ............................................. 97
SubjectPacket ..................................... 89
  USE ............................................. 89
  MAJOR .......................................... 89
  COHERENT ..................................... 90
  TYPES .......................................... 90
  target_group ................................... 90
  name ............................................ 90
  remarks ........................................ 90
  type ............................................ 91
  NON_STANDARD_FORMAT_FIELDS ............... 91
  objects ........................................ 91
  __unicode__ .................................... 91
  getSubjects .................................... 92
  getMandatorySubjects ......................... 92
  Meta ............................................. 92
SubjectPriorKnowledgeRelation ............. 97
  id ............................................. 97
  subject ....................................... 97
  prior_knowledge ................................ 98
  __unicode__ .................................... 98
  Meta ............................................. 98

Detailed documentation for all declarations in this file follows.

4.1.1 Subject

Declaration  Declared in KroketApp/models.py on line 48:
class Subject(models.Model):

Qualified name  KroketApp.models.Subject

Description  An entity which models a Subject. Each subject is identified by the field 'code'. Some helper functions are defined to aid the speed of development. Data can be fetched from the OWIS-database using the biztalk request GeefVakGegevens. Multiple request are actually combined in this database, as the database has for each year a different
CHAPTER 4. PYTHON (DJANGO)

subject, but we have one subject in this database. Year-dependent variables are placed in the
weak-entity CoursePlanning.

We also define some fields to be part of standard-format, these are used frequently. Therefore, we load these fields by default when fetching information.

If you want to fetch all fields at once, you can use Subject.objects.fetch_all().

→ SubjectPriorKnowledgeRelation → SubjectFollowUpRelation → KeywordSubjectRelation
→ CoursePlanning → KroketApp.manager.DeferManager → KroketApp.query.subject.info

4.1.2 Subject.code

Declaration Declared in KroketApp/models.py on line 54:

code = models.CharField(max_length=8,

Qualified name KroketApp.models.Subject.code

Description The subject code of the current subject, in the OWIS-server this is represented
by VakCode. Can be at most 8 characters long, and must not be NULL. Is a primary key. Is a
standard format field.

4.1.3 Subject.name

Declaration Declared in KroketApp/models.py on line 60:

name = models.CharField(max_length=75,

Qualified name KroketApp.models.Subject.name

Description The name of the subject, in the ‘‘GeefVakGegevens’’ request this is represented
by VakOmschr, can be at most 75 characters long, and must not be NULL. Is a standard
format field.

4.1.4 Subject.niveau

Declaration Declared in KroketApp/models.py on line 66:

niveau = models.CharField(max_length=50)

Qualified name KroketApp.models.Subject.niveau

Description The difficulty of the subject, given as Basic, Intermediate or Advanced. In the
‘‘GeefVakGegevens’’ request this is represented by VakNiveau. Is a standard format field.

4.1.5 Subject.ects

Declaration Declared in KroketApp/models.py on line 71:

ects = models.IntegerField(verbose_name='ECTS',
4.1. KROKETAPP.MODELS

Qualified name  KroketApp.models.Subject.ects

Description  The amount ECTS this Subject is worth, as an Integer, must not be NULL, is by default 5. In the ‘‘GeefVakGegevens’’ request this is represented by SpBama. Is a standard format field.

4.1.6  Subject.remarks

Declaration  Declared in KroketApp/models.py on line 78:
remarks = models.TextField(blank=True)

Qualified name  KroketApp.models.Subject.remarks

Description  The remarks given by the Subject. This is a string which can be as large as possible. In the ‘‘GeefVakGegevens’’ request this is represented by Opmerking. Is by default blank. Is not a standard format field.

4.1.7  Subject.study_goal

Declaration  Declared in KroketApp/models.py on line 84:
study_goal = models.TextField(blank=True)

Qualified name  KroketApp.models.Subject.study_goal

Description  The study goal of the Subject. This is a string which can be as large as possible. In the ‘‘GeefVakGegevens’’ request this is represented by Leerdoel. Is by default blank. Is not a standard format field.

4.1.8  Subject.content

Declaration  Declared in KroketApp/models.py on line 90:
content = models.TextField(blank=True)

Qualified name  KroketApp.models.Subject.content

Description  The content of the Subject. This is a string which can be as large as possible. In the ‘‘GeefVakGegevens’’ request this is represented by Inhoud. Is by default blank. Is not a standard format field.

4.1.9  Subject.weekly_content

Declaration  Declared in KroketApp/models.py on line 96:
weekly_content = models.TextField(blank=True)
CHAPTER 4. PYTHON (DJANGO)

**Qualified name**  KroketApp.models.Subject.weekly_content

**Description**  The weekly content of the Subject. This is a string which can be as large as possible. In the ‘‘GeefVakGegevens’’ request this is represented by WeekInhoud. Is by default blank. Is not a standard format field.

### 4.1.10 Subject.last_year

**Declaration**  Declared in KroketApp/models.py on line 103:

```python
last_year = models.IntegerField(default=9999,
```

**Qualified name**  KroketApp.models.Subject.last_year

**Description**  The last year this Subject is given. This is represented by an Integer, which is by default 9999 (indicates unknown). In the ‘‘GeefVakGegevens’’ request this is represented by Studiejaar_Laatst_Geg. In the admin interface this is named ‘Last year given’. Is not a standard format field.

### 4.1.11 Subject.video_mat_descr

**Declaration**  Declared in KroketApp/models.py on line 111:

```python
video_mat_descr = models.CharField(max_length=50, blank=True,
```

**Qualified name**  KroketApp.models.Subject.video_mat_descr

**Description**  The video material description given by the Subject. This is represented by a String of maximum length 50, which is by default blank. In the ‘‘GeefVakGegevens’’ request this is represented by VideoMateriaalOms. In the admin interface this is named ‘Video-Material Description’. Is not a standard format field.

### 4.1.12 Subject.video_url

**Declaration**  Declared in KroketApp/models.py on line 119:

```python
video_url = models.CharField(max_length=100, blank=True,
```

**Qualified name**  KroketApp.models.Subject.video_url

**Description**  The URL to the video material of the Subject. This is represented by a String of maximum length 100, which is by default blank. In the ‘‘GeefVakGegevens’’ request this is represented by VideoMateriaalUrl. In the admin interface this is named ‘Video-Material URL’. Is not a standard format field.
4.1. KROKETAPP.MODELS

4.1.13 Subject.url

Declaration  Declared in KroketApp/models.py on line 127:
url = models.CharField(max_length=100, blank=True),

Qualified name  KroketApp.models.Subject.url

Description  The URL linking to the study planning of the Subject. This is represented by
a string of maximum length 100, which is by default blank. In the ‘‘GeefVakGegevens’’
request this is represented by UrlStudiewijzer. In the admin interface this is named ‘URL’.
Is not a standard format field.

4.1.14 Subject.education_type

Declaration  Declared in KroketApp/models.py on line 135:
education_type = models.CharField(max_length=4000, blank=True) #OnderwijsvormInfo

Qualified name  KroketApp.models.Subject.education_type

Description  A description of the education type of the Subject. This is represented by
a string of maximum length 4000, which is by default blank. In the ‘‘GeefVakGegevens’’
request this is represented by OnderwijsvormInfo. Is not a standard format field.

4.1.15 Subject.examination_type

Declaration  Declared in KroketApp/models.py on line 142:
examination_type= models.CharField(max_length=4000, blank=True)

Qualified name  KroketApp.models.Subject.examination_type

Description  A description of the examination type of the Subject. This is represented by
a string of maximum length 4000, which is by default blank. In the ‘‘GeefVakGegevens’’
request this is represented by ToetsvormInfo. Is not a standard format field.

4.1.16 Subject.department

Declaration  Declared in KroketApp/models.py on line 149:
department = models.CharField(max_length=50, blank=True)

Qualified name  KroketApp.models.Subject.department

Description  The name of the department that is responsible for this Subject. This is repre-
sented by a string of maximum length 50, which is by default blank. In the ‘‘GeefVakGegevens’’
request this is represented by the multi-field verzorgende_eenheden. Is not a standard for-
mat field.
CHAPTER 4. PYTHON (DJANGO)

4.1.17 Subject.subdepartment

Declaration  Declared in KroketApp/models.py on line 156:
subdepartment = models.CharField(max_length=100, blank=True)

Qualified name  KroketApp.models.Subject.subdepartment

Description  The name of the subdepartment that is responsible for this Subject. This
is represented by a string of maximum length 100, which is by default blank. In the
‘GeefVakGegevens’ request this is represented by verzorgende_eenheden. Is not a stan-
dard format field.

4.1.18 Subject.rating

Declaration  Declared in KroketApp/models.py on line 161:
rating = models.FloatField()

Qualified name  KroketApp.models.Subject.rating

Description  The rating of this Subject as given by pollweb.tue.nl. This is a floating point
number between 1 and 10, where 10 is the best possible score and 1 is the worst possible
score. Is not a standard format field.

4.1.19 Subject.NON_STANDARD_FORMAT_FIELDS

Declaration  Declared in KroketApp/models.py on line 166:
NON_STANDARD_FORMAT_FIELDS = ['remarks', 'study_goal', 'content', 'weekly_content',
'video_mat_descr', 'video_url',

Qualified name  KroketApp.models.Subject.NON_STANDARD_FORMAT_FIELDS

Description  The fields that are not in the standard format, and are not normally necessary
to load.

4.1.20 Subject.objects

Declaration  Declared in KroketApp/models.py on line 171:
objects = manager.DeferManager(NON_STANDARD_FORMAT_FIELDS)

Qualified name  KroketApp.models.Subject.objects

Description  The optimized object manager, which only loads default fields in standard
format.
4.1. KROKETAPP.MODELS

4.1.21 Subject.getFollowUp

Declaration  Declared in KroketApp/models.py on line 178:
```
def getFollowUp(self):
```

Qualified name  KroketApp.models.Subject.getFollowUp

Description  Gets all follow-up subjects of this subject.

Parameters

```
self – (Subject)  The Subject on which the function is called.
```

Returns  (Subject-List)  A list of Subject which are follow up subjects to this subject.

4.1.22 Subject.getPriorKnowledge

Declaration  Declared in KroketApp/models.py on line 189:
```
def getPriorKnowledge(self):
```

Qualified name  KroketApp.models.Subject.getPriorKnowledge

Description  Gets all prior knowledge subjects of this subject.

Parameters

```
self – (Subject)  The Subject on which the function is called.
```

Returns  (Subject-List)  A list of subjects which are considered required prior knowledge to this subject.

4.1.23 Subject.__unicode__

Declaration  Declared in KroketApp/models.py on line 199:
```
def __unicode__(self):
```

Qualified name  KroketApp.models.Subject.__unicode__

Description  A description of the subject.

Parameters

```
self – (Subject)  The Subject on which the function is called.
```
CHAPTER 4. PYTHON (DJANGO)

Returns  (String) A description of this subject. This is a String with the subject code and name.

4.1.24 Keyword

Declaration  Declared in KroketApp/models.py on line 212:
class Keyword(models.Model):

Qualified name  KroketApp.models.Keyword

Description  A keyword which is used to describe a SubjectPacket or a Subject. Is used to search on these entities. Is created by the Python-script categorize by parsing certain fields of a SubjectPacket or a Subject. categorize has also found the type (verb, noun, gerund, etc...) of this word. This also gives the keyword a multiplier (as seen in categorize), which is used to determine the importance of the word.

→ KroketApp.categorize→ SubjectPacket→ Subject→ KeywordSubjectRelation → KeywordPacketRelation

4.1.25 Keyword.name

Declaration  Declared in KroketApp/models.py on line 216:
name = models.CharField(max_length=80, primary_key = True)

Qualified name  KroketApp.models.Keyword.name

Description  The actual keyword as string, is a primary key. Can be at most 80 characters long. Must not be NULL.

4.1.26 Keyword.total_count

Declaration  Declared in KroketApp/models.py on line 223:
total_count = models.IntegerField()

Qualified name  KroketApp.models.Keyword.total_count

Description  An integer which has as value the frequency of this keyword found in SubjectPacket or Subject times the type-multiplier. If the word is found in special fields (e.g. subject code, subject name, subject, department) this will count as if the keyword is found a high amount of times in that subject. Must not be NULL

4.1.27 Keyword.__unicode__

Declaration  Declared in KroketApp/models.py on line 230:
def __unicode__(self):
4.1. KROKETAPP.MODELS

Qualified name  KroketApp.models.Keyword.__unicode__

Description  A description of the keyword.

Parameters

  self – (Keyword) The keyword on which the function is called.

Returns  (String) A description of this keyword.

4.1.28  SubjectPacket

Declaration  Declared in KroketApp/models.py on line 247:
  class SubjectPacket(models.Model):

Qualified name  KroketApp.models.SubjectPacket

Description  An entity which models a packet of subjects. These packets can be either of the type USE-package, coherent-package or a major. The type of a packet is found by using the ‘type’ field. These can be of the value ‘SubjectPacket.USE’, ‘SubjectPacket.MAJOR’ or ‘SubjectPacket.COHERENT’. Each packet is identified by the field ‘target_group’. These fields are integers and are taken from the OWIS-database.

  We also define some fields to be part of standard-format, these are used frequently. Therefore, we load these fields by default when fetching information.

  If you want to fetch all fields at once, you can use Subject.objects.fetch_all().
  → KroketApp.manager.DeferManager

4.1.29  SubjectPacket.USE

Declaration  Declared in KroketApp/models.py on line 251:
  USE = 1

Qualified name  KroketApp.models.SubjectPacket.USE

Description  A constant that can be used to identify USE packages using the ‘type’ field.

4.1.30  SubjectPacket.MAJOR

Declaration  Declared in KroketApp/models.py on line 255:
  MAJOR = 2

Qualified name  KroketApp.models.SubjectPacket.MAJOR
CHAPTER 4. PYTHON (DJANGO)

Description  A constant that can be used to identify majors using the ‘type’ field.

4.1.31  SubjectPacket.COHERENT

Declaration  Declared in KroketApp/models.py on line 259:
  COHERENT = 3

Qualified name  KroketApp.models.SubjectPacket.COHERENT

Description  A constant that can be used to identify USE packages using the ‘type’ field.

4.1.32  SubjectPacket.TYPES

Declaration  Declared in KroketApp/models.py on line 263:
  TYPES = ((USE, 'USE'), (MAJOR, 'Major'), (COHERENT,'Coherent'))

Qualified name  KroketApp.models.SubjectPacket.TYPES

Description  A tuple of tuples assigning to each type constant a human-readable string.

4.1.33  SubjectPacket.target_group

Declaration  Declared in KroketApp/models.py on line 269:
  target_group = models.IntegerField(primary_key=True)

Qualified name  KroketApp.models.SubjectPacket.target_group

Description  The target_group identifier (primary key) of the packet as an Integer. Is a standard format field.

4.1.34  SubjectPacket.name

Declaration  Declared in KroketApp/models.py on line 274:
  name = models.CharField(max_length=80)

Qualified name  KroketApp.models.SubjectPacket.name

Description  The name of the package as a string of most 80 characters, must not be NULL. Is a standard format field.

4.1.35  SubjectPacket.remarks

Declaration  Declared in KroketApp/models.py on line 279:
  remarks = models.TextField()
Qualified name  KroketApp.models.SubjectPacket.remarks

Description  The remarks of the package as an arbitrary long string. Is not a standard format field.

4.1.36  SubjectPacket.type

Declaration  Declared in KroketApp/models.py on line 284:
                type = models.SmallIntegerField(choices=TYPES)

Qualified name  KroketApp.models.SubjectPacket.type

Description  The type of the field, which is an integer inside the range given by codeTYPES. Is a standard format field. Must not be NULL.

4.1.37  SubjectPacket.NON_STANDARD_FORMAT_FIELDS

Declaration  Declared in KroketApp/models.py on line 290:
                NON_STANDARD_FORMAT_FIELDS = ['remarks']

Qualified name  KroketApp.models.SubjectPacket.NON_STANDARD_FORMAT_FIELDS

Description  The fields that are not in the standard format, and are not normally necessary to load.

4.1.38  SubjectPacket.objects

Declaration  Declared in KroketApp/models.py on line 294:
                objects = manager.DeferManager(NON_STANDARD_FORMAT_FIELDS)

Qualified name  KroketApp.models.SubjectPacket.objects

Description  The optimized object manager, which only loads by default fields in standard format.

4.1.39  SubjectPacket.__unicode__

Declaration  Declared in KroketApp/models.py on line 301:
                def __unicode__(self):

Qualified name  KroketApp.models.SubjectPacket.__unicode__

Description  The description of the package, which is just the name.
CHAPTER 4. PYTHON (DJANGO)

Parameters

self – (SubjectPacket) The instance of SubjectPacket.

4.1.40  SubjectPacket.getSubjects

Declaration  Declared in KroketApp/models.py on line 311:
def getSubjects(self):

Qualified name  KroketApp.models.SubjectPacket.getSubjects

Description  Fetches the subject which are part of this packet. Subjects are part of a packet if these are scheduled for the subjectPacket in CoursePlanning. → CoursePlanning

Parameters

self – (SubjectPacket) The instance of the SubjectPacket.

Returns  (QuerySet) A QuerySet of subjects, which are part of this package.

4.1.41  SubjectPacket.getMandatorySubjects

Declaration  Declared in KroketApp/models.py on line 323:
def getMandatorySubjects(self):

Qualified name  KroketApp.models.SubjectPacket.getMandatorySubjects

Description  Fetches the subject which are part of this packet. Subjects are part of a packet if these are scheduled for the subjectPacket in CoursePlanning. → CoursePlanning

Parameters

self – (SubjectPacket) The instance of the SubjectPacket.

Returns  (QuerySet) A QuerySet of subjects, which are part of this package.

4.1.42  SubjectPacket.Meta

Declaration  Declared in KroketApp/models.py on line 331:
class Meta:

Qualified name  KroketApp.models.SubjectPacket.Meta
4.1. KROKETAPP.MODELS

Description Defines the verbose name of this class.

4.1.43 CoursePlanning

Declaration Declared in KroketApp/models.py on line 345:
class CoursePlanning(models.Model):

Qualified name KroketApp.models.CoursePlanning

Description A weak-entity that models the information about when a certain subject is scheduled for a certain SubjectPacket. As a packet can be scheduled multiple times, this is not just a relation between a Subject and a SubjectPacket, but rather a weak entity that is identified by the combination of fields: (subject,target_group, year, quartile, timeslot). → Subject→SubjectPacket

4.1.44 CoursePlanning.id

Declaration Declared in KroketApp/models.py on line 350:
id = models.AutoField(primary_key=True)

Qualified name KroketApp.models.CoursePlanning.id

Description The auto-increment id-field that is used as a primary_key for the internal database (we abstract from this key in the comments).

4.1.45 CoursePlanning.subject

Declaration Declared in KroketApp/models.py on line 355:
subject = models.ForeignKey(Subject)

Qualified name KroketApp.models.CoursePlanning.subject

Description A foreign key to the Subject which is scheduled. Must not be NULL.

4.1.46 CoursePlanning.target_group

Declaration Declared in KroketApp/models.py on line 359:
target_group = models.ForeignKey(SubjectPacket)

Qualified name KroketApp.models.CoursePlanning.target_group

Description A foreign key to the SubjectPacket for which the Subject is scheduled. Must not be NULL.
CHAPTER 4. PYTHON (DJANGO)

4.1.47 CoursePlanning.year

Declaration  Declared in KroketApp/models.py on line 364:
year = models.CharField(max_length=50)

Qualified name  KroketApp.models.CoursePlanning.year

Description  An Integer represented as String, representing the year (1-3) in which a student, which is exactly on schedule, would take the course. Must not be NULL.

4.1.48 CoursePlanning.quartile

Declaration  Declared in KroketApp/models.py on line 369:
quartile = models.CharField(max_length=10, blank=True)

Qualified name  KroketApp.models.CoursePlanning.quartile

Description  An Integer represented as String, representing the quartile (1-4) in which the course is given. Is by default blank.

4.1.49 CoursePlanning.timeslot

Declaration  Declared in KroketApp/models.py on line 374:
timeslot = models.CharField(max_length=10, blank=True)

Qualified name  KroketApp.models.CoursePlanning.timeslot

Description  The corresponding timeslot (A-E), separated by semicolons (';'), if the subject has lectures/instructions on multiple timeslots. Must not be NULL and is by default blank.

4.1.50 CoursePlanning.mandatory

Declaration  Declared in KroketApp/models.py on line 378:
mandatory = models.BooleanField()

Qualified name  KroketApp.models.CoursePlanning.mandatory

Description  A boolean representing if the subject is mandatory to the SubjectPacket. Is by default false.

4.1.51 CoursePlanning.__unicode__

Declaration  Declared in KroketApp/models.py on line 385:
def __unicode__(self):
 Qualified name  KroketApp.models.CoursePlanning.__unicode__

 Description  A string describing the relation, includes the subject description, year, timeslot, quartile and the target group description.

 Parameters

 $self$ – (CoursePlanning) The instance of CoursePlanning.

 4.1.52 CoursePlanning.getPlanning

 Declaration  Declared in KroketApp/models.py on line 397:

 def getPlanning(subj_code):

 Qualified name  KroketApp.models.CoursePlanning.getPlanning

 Description  Given a subject code, this method will return a list of dictionaries containing the values year, timeslot and quartile, describing in which year, quartile and timeslot the subject is scheduled.

 Parameters

 subj_code – (String) The subject code of which the planning data is requested.

 Returns  (List of Dict) A list of dictionaries describing how the subject is planned.

 4.1.53 CoursePlanning.Meta

 Declaration  Declared in KroketApp/models.py on line 407:

 class Meta:

 Qualified name  KroketApp.models.CoursePlanning.Meta

 Description  A meta class which describes constraints and other meta-information.

 4.1.54 SubjectFollowUpRelation

 Declaration  Declared in KroketApp/models.py on line 419:

 class SubjectFollowUpRelation(models.Model):

 Qualified name  KroketApp.models.SubjectFollowUpRelation
CHAPTER 4. PYTHON (DJANGO)

**Description**   A relation between a subject to a subject, which is identifies, what subject is a follow up subject to another subject. Has no other useful information. The combination of foreign-keys: (subject, follow_up) is a primary key to this relation. → Subject

### 4.1.55 SubjectFollowUpRelation.id

**Declaration**   Declared in KroketApp/models.py on line 425:
```python
id = models.AutoField(primary_key=True)
```

**Qualified name**   KroketApp.models.SubjectFollowUpRelation.id

**Description**   The auto-increment id-field that is used as a primary_key for the internal database (we abstract from this key in the comments).

### 4.1.56 SubjectFollowUpRelation.subject

**Declaration**   Declared in KroketApp/models.py on line 429:
```python
subject = models.ForeignKey(Subject)
```

**Qualified name**   KroketApp.models.SubjectFollowUpRelation.subject

**Description**   The given subject to which follow_up is a follow up subject. Must not be NULL. → Subject

### 4.1.57 SubjectFollowUpRelation.follow_up

**Declaration**   Declared in KroketApp/models.py on line 433:
```python
follow_up = models.ForeignKey(Subject, related_name='follow_up',
```

**Qualified name**   KroketApp.models.SubjectFollowUpRelation.follow_up

**Description**   The subject which is a follow-up subject to subject. → Subject

### 4.1.58 SubjectFollowUpRelation.__unicode__

**Declaration**   Declared in KroketApp/models.py on line 440:
```python
def __unicode__(self):
```

**Qualified name**   KroketApp.models.SubjectFollowUpRelation.__unicode__

**Description**   A string describing the relation, includes the description of both subject and follow_up.
4.1. KROKETAPP.MODELS

Parameters

self – (SubjectFollowUpRelation) The instance of SubjectFollowUpRelation.

4.1.59 SubjectFollowUpRelation.Meta

Declaration  Declared in KroketApp/models.py on line 446:
class Meta:

Qualified name  KroketApp.models.SubjectFollowUpRelation.Meta

Description  A meta class which describes constraints and other meta-information.

4.1.60 SubjectPriorKnowledgeRelation

Declaration  Declared in KroketApp/models.py on line 454:
class SubjectPriorKnowledgeRelation(models.Model):

Qualified name  KroketApp.models.SubjectPriorKnowledgeRelation

Description  A relation between a subject to a subject, which is identifies, what subject is considered prior knowledge to a subject. Has no other useful information. The combination of foreign-keys: (subject, follow_up) is a primary key to this relation.

4.1.61 SubjectPriorKnowledgeRelation.id

Declaration  Declared in KroketApp/models.py on line 459:
id = models.AutoField(primary_key=True)

Qualified name  KroketApp.models.SubjectPriorKnowledgeRelation.id

Description  The auto-increment id-field that is used as a primary_key for the internal database (we abstract from this key in the comments).

4.1.62 SubjectPriorKnowledgeRelation.subject

Declaration  Declared in KroketApp/models.py on line 465:
subject = models.ForeignKey(Subject)

Qualified name  KroketApp.models.SubjectPriorKnowledgeRelation.subject

Description  The given subject to which prior_knowledge is considered prior knowledge subject. Must not be NULL. → Subject
CHAPTER 4. PYTHON (DJANGO)

4.1.63 SubjectPriorKnowledgeRelation.prior_knowledge

**Declaration**  Declared in KroketApp/models.py on line 469:

```python
prior_knowledge = models.ForeignKey(Subject, related_name='prior_knowledge',
```

**Qualified name**  KroketApp.models.SubjectPriorKnowledgeRelation.prior_knowledge

**Description**  The subject which is considered prior knowledge to subject. → Subject

4.1.64 SubjectPriorKnowledgeRelation.__unicode__

**Declaration**  Declared in KroketApp/models.py on line 476:

```python
def __unicode__(self):
```

**Qualified name**  KroketApp.models.SubjectPriorKnowledgeRelation.__unicode__

**Description**  A string describing the relation, includes the description of both subject and prior_knowledge.

**Parameters**

- `self` – (SubjectPriorKnowledgeRelation) The instance of SubjectPriorKnowledgeRelation.

4.1.65 SubjectPriorKnowledgeRelation.Meta

**Declaration**  Declared in KroketApp/models.py on line 483:

```python
class Meta:
```

**Qualified name**  KroketApp.models.SubjectPriorKnowledgeRelation.Meta

**Description**  A meta class which describes constraints and other meta-information.

4.1.66 KeywordSubjectRelation

**Declaration**  Declared in KroketApp/models.py on line 497:

```python
class KeywordSubjectRelation(models.Model):
```

**Qualified name**  KroketApp.models.KeywordSubjectRelation
4.1.1. KROKETAPP.MODELS

Description  A relation between a subject and a keyword, which is used to describe a Subject. Is used to search on fields in the Subject. Is created by the Python-script categorize by parsing certain fields of a Subject.categorize has also found the type (verb, noun, gerund, etc...) of this word. This also gives the keyword a multiplier (as seen in categorize), which is used to determine the importance of the word. to determine the importance of this keyword.

The combination of foreign-keys: (Keyword, subject) is a primary key to this relation. KroketApp.categorize→Keyword→Subject

4.1.67  KeywordSubjectRelation.id

Declaration  Declared in KroketApp/models.py on line 502:
id = models.AutoField(primary_key=True)

Qualified name  KroketApp.models.KeywordSubjectRelation.id

Description  The auto-increment id-field that is used as a primary_key for the internal database (we abstract from this key in the comments).

4.1.68  KeywordSubjectRelation.keyword

Declaration  Declared in KroketApp/models.py on line 507:
keyword = models.ForeignKey(Keyword)

Qualified name  KroketApp.models.KeywordSubjectRelation.keyword

Description  A foreign key to the keyword which describes the Subject.

4.1.69  KeywordSubjectRelation.subject

Declaration  Declared in KroketApp/models.py on line 512:
subject = models.ForeignKey(Subject)

Qualified name  KroketApp.models.KeywordSubjectRelation.subject

Description  The Subject that is described by the keyword.

4.1.70  KeywordSubjectRelation.count

Declaration  Declared in KroketApp/models.py on line 520:
count = models.IntegerField(default=0)

Qualified name  KroketApp.models.KeywordSubjectRelation.count
**CHAPTER 4. PYTHON (DJANGO)**

**Description**  An integer which has as value the frequency of this keyword found is in the corresponding Subject times the type-multiplier. If the word is found in special fields (e.g. subject code, subject name, subject, department) this will count as if the keyword is found a high amount of times in that subject. Must not be NULL.

### 4.1.71 KeywordSubjectRelation.__unicode__

**Declaration**  Declared in KroketApp/models.py on line 528:

```python
def __unicode__(self):
```

**Qualified name**  KroketApp.models.KeywordSubjectRelation.__unicode__

**Description**  A string describing the relation, includes the description of both subject and keyword and also has the value of the count.

**Parameters**

- `self` – (KeywordSubjectRelation) The instance of KeywordSubjectRelation.

**Returns**  (String) A description of this relation.

### 4.1.72 KeywordSubjectRelation.Meta

**Declaration**  Declared in KroketApp/models.py on line 534:

```python
class Meta:
```

**Qualified name**  KroketApp.models.KeywordSubjectRelation.Meta

**Description**  A meta class which describes constraints and other meta-information.

### 4.1.73 KeywordPacketRelation

**Declaration**  Declared in KroketApp/models.py on line 547:

```python
class KeywordPacketRelation(models.Model):
```

**Qualified name**  KroketApp.models.KeywordPacketRelation

**Description**  A relation between a SubjectPacket and a keyword, which is used to describe a SubjectPacket. Is used to search on fields in the Subject. Is created by the Python-script categorize by parsing certain fields of a SubjectPacket.categorize has also found the type (verb, noun, gerund, etc...) of this word. This also gives the keyword a multiplier (as seen in categorize), which is used to determine the importance of the word. to determine the importance of this keyword. The combination of foreign-keys: (Keyword, target_group) is a primary key to this relation. → KroketApp.categorize→ Keyword→ Subject
4.1.74 KeywordPacketRelation.id

Declaration  Declared in KroketApp/models.py on line 552:
  id = models.AutoField(primary_key=True)

Qualified name  KroketApp.models.KeywordPacketRelation.id

Description  The auto-increment id-field that is used as a primary_key for the internal database (we abstract from this key in the comments).

4.1.75 KeywordPacketRelation.keyword

Declaration  Declared in KroketApp/models.py on line 557:
  keyword = models.ForeignKey(Keyword)

Qualified name  KroketApp.models.KeywordPacketRelation.keyword

Description  A foreign key to the keyword which describes the SubjectPacket.

4.1.76 KeywordPacketRelation.target_group

Declaration  Declared in KroketApp/models.py on line 561:
  target_group = models.ForeignKey(SubjectPacket)

Qualified name  KroketApp.models.KeywordPacketRelation.target_group

Description  The SubjectPacket that is described by the keyword.

4.1.77 KeywordPacketRelation.count

Declaration  Declared in KroketApp/models.py on line 569:
  count = models.IntegerField(default=0)

Qualified name  KroketApp.models.KeywordPacketRelation.count

Description  An integer which has as value the frequency of this keyword found is in the corresponding Subject times the type-multiplier. If the word is found in special fields (e.g. subject code, subject name, subject, department) this will count as if the keyword is found a high amount of times in that subject. Must not be NULL.

4.1.78 KeywordPacketRelation.__unicode__

Declaration  Declared in KroketApp/models.py on line 577:
  def __unicode__(self):
CHAPTER 4. PYTHON (DJANGO)

Qualified name  KroketApp.models.KeywordPacketRelation.__unicode__

Description  A string describing the relation, includes the description of both subject and keyword and also has the value of the count.

Parameters

self – (KeywordSubjectRelation) The instance of KeywordSubjectRelation.

Returns  (String) A description of this relation.

4.1.79  KeywordPacketRelation.Meta

Declaration  Declared in KroketApp/models.py on line 583:

class Meta:

Qualified name  KroketApp.models.KeywordPacketRelation.Meta

Description  A meta class which describes constraints and other meta-information.
4.2 KroketApp.admin

The admin.py script, that Django uses to create the admin-interface. More information about the models can be found in models.py.

Hierarchical member index    This file contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountAdmin</td>
<td>106</td>
</tr>
<tr>
<td>CPInline</td>
<td>104</td>
</tr>
<tr>
<td>KWPInline</td>
<td>104</td>
</tr>
<tr>
<td>KWSInline</td>
<td>104</td>
</tr>
<tr>
<td>KeywordAdmin</td>
<td>106</td>
</tr>
<tr>
<td>RelatedSubjectAdmin</td>
<td>107</td>
</tr>
<tr>
<td>SFURInline</td>
<td>103</td>
</tr>
<tr>
<td>SPCPInline</td>
<td>104</td>
</tr>
<tr>
<td>SPKRInline</td>
<td>103</td>
</tr>
<tr>
<td>ScheduleAdmin</td>
<td>106</td>
</tr>
<tr>
<td>ScheduleInline</td>
<td>105</td>
</tr>
<tr>
<td>ScheduleSubjectAdmin</td>
<td>106</td>
</tr>
<tr>
<td>ScheduleSubjectRelInline</td>
<td>105</td>
</tr>
<tr>
<td>SubjectAdmin</td>
<td>105</td>
</tr>
<tr>
<td>SubjectPacketAdmin</td>
<td>105</td>
</tr>
</tbody>
</table>

Detailed documentation for all declarations in this file follows.

4.2.1 SFURInline

Declaration    Declared in KroketApp/admin.py on line 22:

```python
class SFURInline(admin.TabularInline):
```

Qualified name    KroketApp.admin.SFURInline

Description    An inline admin class which extends the admin.TabularInline class, and by using that class, directs requests to the ‘objects’ database. Using this class, the database model SubjectFollowUpRelation can be modified.

4.2.2 SPKRInline

Declaration    Declared in KroketApp/admin.py on line 34:

```python
class SPKRInline(admin.TabularInline):
```

Qualified name    KroketApp.admin.SPKRInline

Description    An inline admin class which extends the admin.TabularInline class, and by using that class, directs requests to the ‘objects’ database. Using this class, the database model SubjectPriorKnowledgeRelation can be modified.
CHAPTER 4. PYTHON (DJANGO)

4.2.3 KWSInline

Declaration  Declared in KroketApp/admin.py on line 46:

```python
class KWSInline(admin.TabularInline):
```

Qualified name  KroketApp.admin.KWSInline

Description  An inline admin class which extends the admin.TabularInline class, and by using that class, directs requests to the ‘objects’ database. Using this class, the database model KeywordSubjectRelation can be modified.

4.2.4 KWPInline

Declaration  Declared in KroketApp/admin.py on line 57:

```python
class KWPInline(admin.TabularInline):
```

Qualified name  KroketApp.admin.KWPInline

Description  An inline admin class which extends the admin.TabularInline class, and by using that class, directs requests to the ‘objects’ database. Using this class, the database model KeywordPacketRelation can be modified.

4.2.5 CPInline

Declaration  Declared in KroketApp/admin.py on line 68:

```python
class CPInline(admin.TabularInline):
```

Qualified name  KroketApp.admin.CPInline

Description  An inline admin class which extends the admin.TabularInline class, and by using that class, directs requests to the ‘objects’ database. Using this class, the database model CoursePlanning can be modified. Is different from SPCPInline, because the fields are tabular in the admin interface.

4.2.6 SPCPInline

Declaration  Declared in KroketApp/admin.py on line 79:

```python
class SPCPInline(admin.StackedInline):
```

Qualified name  KroketApp.admin.SPCPInline

Description  An inline admin class which extends the admin.StackedInline class, and by using that class, directs requests to the ‘objects’ database. Using this class, the database model CoursePlanning can be modified. Is different from CPInline, because the fields are stacked in the admin interface.
4.2.7 ScheduleSubjectRelInline

Declaration  Declared in KroketApp/admin.py on line 89:
class ScheduleSubjectRelInline(admin.StackedInline):

Qualified name  KroketApp.admin.ScheduleSubjectRelInline

Description  An inline admin class which extends the StackedInline class, and by using
that class, directs requests to the default ('user') database. Using the class, the database
model ScheduleSubjectRelation can be modified.

4.2.8 ScheduleInline

Declaration  Declared in KroketApp/admin.py on line 99:
class ScheduleInline(admin.TabularInline):

Qualified name  KroketApp.admin.ScheduleInline

Description  An inline admin class which extends the StackedInline class, and by using
that class, directs requests to the default ('user') database. Using the class, the database
model AccountScheduleRelation can be modified.

4.2.9 SubjectPacketAdmin

Declaration  Declared in KroketApp/admin.py on line 117:
class SubjectPacketAdmin(admin.ModelAdmin):

Qualified name  KroketApp.admin.SubjectPacketAdmin

Description  An admin interface which extends the MultiDBModelAdmin class, and by using
that class, directs requests to the 'objects' database. Using the class, the database model
SubjectPacket can be modified. Has as inline interface SPCPInline (CoursePlanning inline)

4.2.10 SubjectAdmin

Declaration  Declared in KroketApp/admin.py on line 125:
class SubjectAdmin(admin.ModelAdmin):

Qualified name  KroketApp.admin.SubjectAdmin

Description  An admin interface which extends the MultiDBModelAdmin class, and by using
that class, directs requests to the 'objects' database. Using the class, the database model
Subject can be modified. Has as inline interface SFURInline (follow-up subjects),
SPKRIInline (prior knowledge subjects), CPInline (course plannings) and KWSInline (keywords).
CHAPTER 4. PYTHON (DJANGO)

4.2.11 ScheduleSubjectAdmin

Declaration  Declared in KroketApp/admin.py on line 142:
class ScheduleSubjectAdmin(admin.ModelAdmin):

Qualified name  KroketApp.admin.ScheduleSubjectAdmin

Description  An admin interface which extends the ModelAdmin class, and by using that
class, directs requests to the default ('user') database. Using the class, the database model
ScheduleSubject can be modified.

4.2.12 ScheduleAdmin

Declaration  Declared in KroketApp/admin.py on line 150:
class ScheduleAdmin(admin.ModelAdmin):

Qualified name  KroketApp.admin.ScheduleAdmin

Description  An admin interface which extends the ModelAdmin class, and by using that
class, directs requests to the default ('user') database. Using the class, the database model
Schedule can be modified. Has as inline interface ScheduleSubjectRelInline (the sched-
uled subjects)

4.2.13 KeywordAdmin

Declaration  Declared in KroketApp/admin.py on line 158:
class KeywordAdmin(admin.ModelAdmin):

Qualified name  KroketApp.admin.KeywordAdmin

Description  An admin interface which extends the MultiDBModelAdmin class, and by using
that class, directs requests to the 'objects' database. Using the class, the database model
Keyword can be modified. Has as inline interface KWSInline, which allows you to view the
related subjects.

4.2.14 AccountAdmin

Declaration  Declared in KroketApp/admin.py on line 166:
class AccountAdmin(admin.ModelAdmin):

Qualified name  KroketApp.admin.AccountAdmin

Description  An admin interface which extends the ModelAdmin class, and by using that
class, directs requests to the default ('user') database. Using the class, the database model
Schedule can be modified. Has as inline interface ScheduleInline (the saved schedules).
4.2. KROKETAPP.ADMIN

4.2.15 RelatedSubjectAdmin

Declaration  Declared in KroketApp/admin.py on line 173:
class RelatedSubjectAdmin(admin.ModelAdmin):

Qualified name  KroketApp.admin.RelatedSubjectAdmin

Description  An admin interface which extends the ModelAdmin class, and by using that class, directs requests to the default ('user') database. Using the class, the database model RelatedSubject can be modified.
CHAPTER 4. PYTHON (DJANGO)

4.3 KroketApp.categorize

A Python script containing declarations of several functions that are able to analyse subjects in the database (see models.py), and generates keywords with weights. It is built on NLTK (the Natural Language Toolkit, http://nltk.org/) and Wordnet (http://wordnet.princeton.edu/), to be able to find the keywords in the description of the subject. The main method, that is able to do this, is rankEnglish.

At the moment only ranking of English subjects is supported. Support for generating keywords for packages and more is not included.

Hierarchical member index  This file contains the following members:

```python
MLStripper .................................................. 108
__init__ .................................................... 109
handle_data ............................................. 109
get_data .................................................... 109
allowed ..................................................... 112
dutch_stopwords ........................................ 111
english_stopwords ........................................ 111
lmtzr ....................................................... 110
multiplier .................................................. 112
rankEnglish ............................................... 114
rankPacket ............................................... 114
rankStringNLTK .......................................... 113
rankSubject .............................................. 113
strip_accents ............................................ 110
strip_tags .................................................. 110
updatePacketDB ........................................... 115
updateSubDB ............................................... 113
wordnet_dict ............................................. 112
```

Detailed documentation for all declarations in this file follows.

4.3.1 MLStripper

Declaration  Declared in KroketApp/categorize.py on line 23:
```python
class MLStripper(HTMLParser):
```

Qualified name  KroketApp.categorize.MLStripper

Description  A class used to strip all HTML tags from a string. This is done by implementing a HTML parser and only appending the data inside each tag to an internal variable fed.
4.3. KROKETAPP.CATEGORIZE

4.3.2 MLStripper.__init__

Declaration  Declared in KroketApp/categorize.py on line 28:
def __init__(self):

Qualified name  KroketApp.categorize.MLStripper.__init__

Description  The init functions: resets the HTMLParser.

Parameters

self – (MLStripper) The instance of this MLStripper.

4.3.3 MLStripper.handle_data

Declaration  Declared in KroketApp/categorize.py on line 38:
def handle_data(self, content):

Qualified name  KroketApp.categorize.MLStripper.handle_data

Description  This method is called to process arbitrary data (e.g. text nodes and the content of <script>...</script> and <style>...<style>).

Parameters

self – (MLStripper) The instance of this MLStripper.
content – (String) The content of the tags as in a String.

4.3.4 MLStripper.get_data

Declaration  Declared in KroketApp/categorize.py on line 47:
def get_data(self):

Qualified name  KroketApp.categorize.MLStripper.get_data

Description  This method returns all content of all tags concatenated as in the order given by the fed data.

Parameters

self – (MLStripper) The instance of this MLStripper.

Returns  (String) A String containing the content of the data which has been fed to this MLStripper.
CHAPTER 4. PYTHON (DJANGO)

4.3.5  strip_tags

Declaration  Declared in KroketApp/categorize.py on line 55:
def strip_tags(html):

Qualified name  KroketApp.categorize.strip_tags

Description  Strips the tags from the given HTML string.

Parameters

  HTML – (String) The HTML content.

Returns  (String) The content without HTML tags.

4.3.6  strip_accents

Declaration  Declared in KroketApp/categorize.py on line 68:
def strip_accents(s):

Qualified name  KroketApp.categorize.strip_accents

Description  A helper method that strips all accents from a Unicode string. If the string is not Unicode, nothing is done.

Parameters

  s – (ASCII/Unicode string) The string to be stripped from accents.

Returns  (ASCII/Unicode string) The stripped string.

4.3.7  lmtzr

Declaration  Declared in KroketApp/categorize.py on line 79:
lmtzr = WordNetLemmatizer()

Qualified name  KroketApp.categorize.lmtzr

Description  The global Lemmatizer, that is used to convert words into their basic form (e.g. ‘walking’ → ‘walk’). → nltk.stem.wordnet.
4.3. KROKETAPP.CATEGORIZE

4.3.8 english_stopwords

Declaration  Declared in KroketApp/categorize.py on line 84:

```python
english_stopwords = ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves'
, 'you', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself'
, 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', '
their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that'
, 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being'
, 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', '
the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at'
, 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during'
, 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out'
, 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', '
there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few'
, 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own'
, 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don'
, 'should', 'now']
```

Qualified name  KroketApp.categorize.english_stopwords

Description  A list of English stopwords. These words are used to filter the most common words out of the descriptions.

4.3.9 dutch_stopwords

Declaration  Declared in KroketApp/categorize.py on line 89:

```python
dutch_stopwords = ['de', 'en', 'van', 'ik', 'te', 'dat', 'die', 'in', 'een', '
hij', 'het', 'niet', 'zijn', 'is', 'was', 'op', 'aan', 'met', 'als', 'voor'
, 'had', 'er', 'maar', 'om', 'hem', 'dan', 'zou', 'of', 'wat', 'mijn', 'men'
, 'dit', 'zo', 'door', 'over', 'ze', 'zich', 'bij', 'ook', 'tot', 'je', 'mijn'
, 'uit', 'der', 'daar', 'haar', 'naar', 'heb', 'hoe', 'heeft', 'hebben', 'deze'
, 'u', 'want', 'nog', 'zal', 'me', 'zij', 'nu', 'ge', 'geen', 'omdat', 'iets'
, 'worden', 'toch', 'al', 'waren', 'veel', 'meer', 'doen', 'toen', 'moet', '
ben', 'zonder', 'kan', 'hun', 'dus', 'alles', 'onder', 'ja', 'eens', 'hier'
, 'wie', 'werd', 'altijd', 'doch', 'wordt', 'wezen', 'kunnen', 'ons', 'zelf'
, 'tegen', 'na', 'reeds', 'wil', 'kon', 'niets', 'uw', 'iemand', 'geweest', '
andere']
```

Qualified name  KroketApp.categorize.dutch_stopwords

Description  A list of Dutch stopwords. These words are used to filter the most common words out of the descriptions.
4.3.10 allowed

Declaration  Declared in KroketApp/categorize.py on line 108:
allowed = ["FW", "NN", "NNS", "NNP", "NNPS", "VBG", "JJ"]

Qualified name  KroketApp.categorize.allowed

Description  The allowed types of a keyword. They can be
   FW foreign word
   NN noun
   NNS plural noun
   NNP proper noun
   NNPS plural proper noun
   VBG verb, gerund or present participle
   JJ adjective
These abbreviations correspond to NLTK’s.

4.3.11 wordnet_dict

Declaration  Declared in KroketApp/categorize.py on line 113:
wordnet_dict = [ n, n, n, n, n, v, aj]

Qualified name  KroketApp.categorize.wordnet_dict

Description  Used to convert the abbreviations from NLTK to Wordnet abbreviations. → allowed

4.3.12 multiplier

Declaration  Declared in KroketApp/categorize.py on line 118:
multiplier = [ 10, 10, 10, 10, 10, 5, 3]

Qualified name  KroketApp.categorize.multiplier

Description  Indicates the weight of a certain word-type. For example a noun has a greater weight than an adjective. → allowed

4.3.13 rankSubject

Declaration  Declared in KroketApp/categorize.py on line 134:
def rankSubject(subject):
Qualified name  KroketApp.categorize.rankSubject

Description  Removes all keywords related to the subject, and generates new keywords by parsing the remarks, study_goal, content and weekly_content fields of subject. This is done by using the rankStringNLTK method. Furthermore, the subject code is also generated as keyword with ranking 200, and the (sub-)department is generated as keyword with ranking 50. The subject name is also split and filtered on stopwords; each part of the name is given a ranking of 200.

The subject should be in the objects database, otherwise the database could get into an inconsistent state. The Keywords and KeywordSubjectsRelations are also added in the objects database.

All HTML tags in the remarks, etc are stripped before generating keywords.

Parameters

subject – (Subject) The subject to be be ranked.

4.3.14  rankStringNLTK

Declaration  Declared in KroketApp/categorize.py on line 191:
def rankStringNLTK(d, input):

Qualified name  KroketApp.categorize.rankStringNLTK

Description  Generates keywords from the string input. The keywords are added to the dictionary d and the ranking is updated appropriately. Keywords are found by parsing the sentence and only accepting words which have an allowed type. \( \rightarrow \) allowed

For every keyword \( k \) found in the string input we get

\[ d[k] := d[k] + \text{count}_{k \in \text{input}} \cdot \text{(weight of } k\text{'s type)} \]

These weights are defined by a global parameter. \( \rightarrow \) multiplier.

Parameters

\( d \) – (Dictionary) The dictionary to be updated.

\( \text{input} \) – (ASCII/Unicode string) The input to be searched for keywords.

4.3.15  updateSubDB

Declaration  Declared in KroketApp/categorize.py on line 221:
def updateSubDB(subject, dict):

Qualified name  KroketApp.categorize.updateSubDB
CHAPTER 4. PYTHON (DJANGO)

Description  Inserts the keywords given by the dictionary dict in the database and associates them with subject and gives this association the appropriate weight.

Parameters

subject – (Subject) The subject to which the keywords are associates.
dict – (Dictionary) The dictionary which gives each keyword k a weight.

4.3.16 rankEnglish

Declaration  Declared in KroketApp/categorize.py on line 245:
def rankEnglish(subject):

Qualified name  KroketApp.categorize.rankEnglish

Description  Removes all keywords related to the subject, and generates new keywords by parsing the remarks, study_goal, content and weekly_content fields of subject. This is done by using the rankStringNLTK method. Furthermore, the subject code is also generated as keyword with ranking 200, and the (sub-)department is generated as keyword with ranking 50. The subject name is also split and filtered on stopwords; each part of the name is given a ranking of 200.

The subject should be in the objects database, otherwise the database could get into an inconsistent state. The Keywords and KeywordSubjectsRelations are also added in the objects database.

All HTML tags in the remarks, etc are stripped before generating keywords.

Parameters

subject – (Subject) The subject to be be ranked.

4.3.17 rankPacket

Declaration  Declared in KroketApp/categorize.py on line 258:
def rankPacket(packet):

Qualified name  KroketApp.categorize.rankPacket

Description  Removes all keywords related to the packet, and generates new keywords by parsing the remarks field of packet. Keywords from the subjects this packet contains are also added. This is done by using the rankStringNLTK method.

The packet should be in the objects database, otherwise the database could get into an inconsistent state. The Keywords and KeywordPacketRelations are also added in the objects database.
4.3. KROKETAPP.CATEGORIZE

Parameters

packet – (SubjectPacket) The subject to be be ranked.

4.3.18 updatePacketDB

Declaration  Declared in KroketApp/categorize.py on line 293:
def updatePacketDB(packet, dict):

Qualified name  KroketApp.categorize.updatePacketDB

Description  Inserts the keywords given by the dictionary dict in the database and asso-
ciates them with packet and gives this association the appropriate weight.

Parameters

packet – (SubjectPacket) The subject to which the keywords are associates.
dict – (Dictionary) The dictionary which gives each keyword k a weight.
CHAPTER 4. PYTHON (DJANGO)

4.4 KroketApp.savemodels

A Python script that defines the database structure of the accounts and saved schedules. This script should be included by models.py to be synced into the SQL database. This file also includes accounts with hashes and passwords. In the future, we might want to use the NT-authentication system; this would deprecate certain fields of Account.

→ Account→ KroketApp.models

In the file an entity ScheduleSubject exists, which has as only field the subject-code. We have chosen not to use Subject as these are deleted and added every day, which would lead to integrity faults. There are, however, some problems which can arise due to this and for which no solution is implemented. It is possible to have a ScheduleSubject exist without having a corresponding Subject, as the OWIS database can delete a Subject. Requests which would fetch information about that subject would then raise 403 errors. As more problems may arise when the OWIS-database simply removes subjects, we have assumed this would not happen. The major associated with a schedule has the same problem. → ScheduleSubjectKroketApp.models.Subject

The models defined in this Python script are either entities, weak-entities or relations. As a reminder, entities are models which can be identified using their own (non-foreign) fields; weak-entities can only by a combination of their own fields and foreign-fields; Relations are primarily build of foreign keys and create relations between (weak-)entities.

In the current state, the models only support one language (English). In the future some fields might be changed into one-to-many relation, to support multiple languages. (A field f from model m will then be a weak-entity with field f, a foreign key to m and a language identifier.) But again, this is not yet supported.

Django does not support primary-key-sets, but always needs a one-field primary key. This would imply that all our models would be entities, and no relations or weak-entities are allowed to exist. To abstract from certain ‘id’ fields, we define in the comments what the primary key-set is and enforce this by adding ‘unique’ constraints.

Hierarchical member index This file contains the following members:

Account ...................................................... 117
CHAR_SET ................................................. 117
SALT_LENGTH ........................................... 117
HASH_TIMES .............................................. 118
name ....................................................... 118
password ............................................... 118
salt ......................................................... 118
register ................................................... 119
changePassword ......................................... 119
hashPassword ........................................... 119
check_password ......................................... 120
__unicode__ ............................................. 120
Schedule .................................................. 121
id ........................................................... 122
Detailed documentation for all declarations in this file follows.

### 4.4.1 Account

**Declaration** Declared in KroketApp/savemodels.py on line 48:
```python
class Account(models.Model):
```

**Qualified name** KroketApp.savemodels.Account

**Description** An entity that models an account. Has a name, a (sha512-)hashed salted password and the corresponding salt. Has as primary key name

### 4.4.2 Account.CHAR_SET

**Declaration** Declared in KroketApp/savemodels.py on line 53:
```python
CHAR_SET = string.ascii_letters + string.punctuation + string.digits
```

**Qualified name** KroketApp.savemodels.Account.CHAR_SET

**Description** A string with all characters that may be used to generate a random salt. These are all ASCII letters, all punctuation symbols and all decimal digits.

### 4.4.3 Account.SALT_LENGTH

**Declaration** Declared in KroketApp/savemodels.py on line 58:
```python
SALT_LENGTH = 100
```

**Qualified name** KroketApp.savemodels.Account.SALT_LENGTH
CHAPTER 4. PYTHON (DJANGO)

Description  The length of each salt.

4.4.4  Account.HASH_TIMES

Declaration  Declared in KroketApp/savemodels.py on line 66:
HASH_TIMES = 1000

Qualified name  KroketApp.savemodels.Account.HASH_TIMES

Description  Amount of hash-iterations performed on the password. Used to slow down a
brute-forcing approach to crack the password-hash. See also: http://en.wikipedia.org/
wiki/Key_stretching. Should be as high as possible without causing significant delay
(±100ms per hashPassword is a good guideline).

4.4.5  Account.name

Declaration  Declared in KroketApp/savemodels.py on line 76:
name = models.CharField(max_length=20, primary_key=True)

Qualified name  KroketApp.savemodels.Account.name

Description  The name of the account, which can be at most 20 characters long. Must not be
NULL. No further constraint are formulated on database level. In /studieplanner/query/user/register
other constraints are formulated and checked. → KroketApp.query.user.authenticate

4.4.6  Account.password

Declaration  Declared in KroketApp/savemodels.py on line 87:
password = models.CharField(max_length=512)

Qualified name  KroketApp.savemodels.Account.password

Description  A SHA512 hash of the salted password as String in hexadecimal form. This
hash is 512 characters long. The hash of a password can be found by the equation:

\[
\text{hash} = \text{sha512}(\text{password} + \text{salt}) \text{ as HexString}
\]

No further constraints are formulated on database level. In /studieplanner/query/user/register
other constraints are formulated and checked. → KroketApp.query.user.authenticate

4.4.7  Account.salt

Declaration  Declared in KroketApp/savemodels.py on line 93:
salt = models.CharField(max_length=SALT_LENGTH)
Qualified name  KroketApp.savemodels.Account.salt

Description  The salt, which consists of SALT_LENGTH random characters out of the set CHAR_SET. Every time the password is changed this salt should be regenerated. → CHAR_SET → SALT_LENGTH

4.4.8 Account.register

Declaration  Declared in KroketApp/savemodels.py on line 106:
def register(name, pw):

Qualified name  KroketApp.savemodels.Account.register

Description  Creates an account from a name and password. Trailing spaces are removed from name. Will raise an error if the name already exists. Unicode is also not supported. No further constraints on username and passwords are formulated in this functions, constraints are formulated and maintained in /studieplanner/query/user/register.

Parameters

name – (String) The requested name of the account.
pw – (String) The requested password of the account.

Returns  (Account) An account acc with acc.name = name and acc.check_password(pw).

4.4.9 Account.changePassword

Declaration  Declared in KroketApp/savemodels.py on line 128:
def changePassword(self, password):

Qualified name  KroketApp.savemodels.Account.changePassword

Description  Changes the password of the account. Will generate a new salt.

Parameters

self – (Account) An instance of the account.
password – (String) The new password of the account.

4.4.10 Account.hashPassword

Declaration  Declared in KroketApp/savemodels.py on line 138:
def hashPassword(password, salt):

119
CHAPTER 4. PYTHON (DJANGO)

Qualified name KroketApp.savemodels.Account.hashPassword

Description Calculates the hash of the password. Uses iteration to slow down a bruteforce attack.

4.4.11 Account.check_password

Declaration Declared in KroketApp/savemodels.py on line 151:
def check_password(self, password):

Qualified name KroketApp.savemodels.Account.check_password

Description Checks if the password is correct (assuming no hash collisions). This is done by checking if the SHA512 hash of password + self.salt corresponds to the password hash.

Parameters

self – (Account) An instance of the account.
password – (String) The password of the user.

Returns (Boolean) true if the password corresponds to the user, otherwise false.

4.4.12 Account.__unicode__

Declaration Declared in KroketApp/savemodels.py on line 159:
def __unicode__(self):

Qualified name KroketApp.savemodels.Account.__unicode__

Description The description of the account as String. This is equal to the name of the account.

Parameters

self – (Account) An instance of the account.

Returns (String) Description of the account.

4.4.13 ScheduleSubject

Declaration Declared in KroketApp/savemodels.py on line 172:
class ScheduleSubject(models.Model):
Qualified name  KroketApp.savemodels.ScheduleSubject

Description  An entity modeling a Subject. Has only field the subject-code is defined. We have chosen to not use Subject as these are deleted and added every day, which would lead to integrity faults. There are, however, some problems which can arise due to this and for which no solution is implemented. It is possible to have a ScheduleSubject exist without having a corresponding Subject, as the OWIS database can delete a Subject. Requests which would fetch information about that subject would then raise 403 errors. As more problems may arise when the OWIS database simply removes subjects, we have assumed this would not happen. → KroketApp.models.Subject

4.4.14  ScheduleSubject.code

Declaration  Declared in KroketApp/savemodels.py on line 177:
\[
\text{code} = \text{models.CharField(max_length=8, primary_key=True)}
\]

Qualified name  KroketApp.savemodels.ScheduleSubject.code

Description  The subject code of the current subject, in the OWIS server this is represented by VakCode. This be at most 8 characters long, and must not be NULL. It is a primary key, and a standard format field.

4.4.15  ScheduleSubject.__unicode__

Declaration  Declared in KroketApp/savemodels.py on line 184:
\[
def \text{__unicode__}(\text{self}):
\]

Qualified name  KroketApp.savemodels.ScheduleSubject.__unicode__

Description  The description of the subject as String. Is equal to the code of the subject

Parameters

\[
\text{self} \rightarrow (\text{ScheduleSubject}) \text{ An instance of the subject.}
\]

Returns  (String) Description of the subject.

4.4.16  Schedule

Declaration  Declared in KroketApp/savemodels.py on line 194:
\[
class \text{Schedule}(\text{models.Model}):
\]

Qualified name  KroketApp.savemodels.Schedule
CHAPTER 4. PYTHON (DJANGO)

Description  A weak-entity which models a schedule. Has a foreign-key to an account, to which the schedule is belonging. Each account can only have one schedule of a certain name, as such the weak-entity is identified by: (account, name).

4.4.17  Schedule.id

Declaration  Declared in KroketApp/savemodels.py on line 199:
id = models.AutoField(primary_key=True)

Qualified name  KroketApp.savemodels.Schedule.id

Description  The auto-increment id-field that is used as a primary_key for the internal database (we abstract from this key in the comments).

4.4.18  Schedule.account

Declaration  Declared in KroketApp/savemodels.py on line 203:
account = models.ForeignKey(Account)

Qualified name  KroketApp.savemodels.Schedule.account

Description  The account to which the schedule belongs. Must not be NULL. Is part of the primary-key set.

4.4.19  Schedule.name

Declaration  Declared in KroketApp/savemodels.py on line 210:
name = models.CharField(max_length=80)

Qualified name  KroketApp.savemodels.Schedule.name

Description  The name of the schedule. Must not be NULL. Is part of the primary-key set. Can be at most 80 characters long. No further constraints are defined on database level. In /studieplanner/query/schedule/save other constraints are formulated and checked. → KroketApp.query.schedule.save

4.4.20  Schedule.major

Declaration  Declared in KroketApp/savemodels.py on line 216:
major = models.IntegerField()

Qualified name  KroketApp.savemodels.Schedule.major
4.4. KROKETAPP.SAVEMODELS

Description  The target_group identifier. Doesn’t have the foreign key constraints, and therefore a corresponding major doesn’t have to exists. It is assumed that this is not the case. This is due to the scheduled job, which empties the database every day, which would lead to integrity errors. Must not be NULL.

4.4.21  Schedule.begin_year

Declaration  Declared in KroketApp/savemodels.py on line 221:
begin_year = models.IntegerField()

Qualified name  KroketApp.savemodels.Schedule.begin_year

Description  The year in which the first year of the schedule start as an Integer. Must not be NULL. If the starting academic year is yyyy-xxxx, the begin_year is defined as yyyy.

4.4.22  Schedule.__unicode__

Declaration  Declared in KroketApp/savemodels.py on line 228:
def __unicode__(self):

Qualified name  KroketApp.savemodels.Schedule.__unicode__

Description  The description of the schedule as a String. Contains the account description and the major.

Parameters

self – (Schedule) An instance of the schedule.

Returns  (String) Description of the schedule.

4.4.23  Schedule.Meta

Declaration  Declared in KroketApp/savemodels.py on line 234:
class Meta:

Qualified name  KroketApp.savemodels.Schedule.Meta

Description  A meta class which describes constraints and other meta-information.

4.4.24  ScheduleSubjectRelation

Declaration  Declared in KroketApp/savemodels.py on line 246:
class ScheduleSubjectRelation(models.Model):
4.25 ScheduleSubjectRelation.schedule

Declaration  Declared in KroketApp/savemodels.py on line 250:
schedule = models.ForeignKey(Schedule)

Qualified name  KroketApp.savemodels.ScheduleSubjectRelation.schedule

Description  The schedule to which the subject belongs. Must not be NULL.

4.26 ScheduleSubjectRelation.subject

Declaration  Declared in KroketApp/savemodels.py on line 254:
schedule = models.ForeignKey(ScheduleSubject)

Qualified name  KroketApp.savemodels.ScheduleSubjectRelation.subject

Description  The subject which is contained in the schedule. Must not be NULL.

4.27 ScheduleSubjectRelation.time_block

Declaration  Declared in KroketApp/savemodels.py on line 266:
time_block = models.IntegerField()

Qualified name  KroketApp.savemodels.ScheduleSubjectRelation.time_block

Description  The amount of quartiles since the first quartile of the begin_year. The time_block starts at 1, which indicates the first quartile of the begin_year of the account. Then time_block increases by each quartile that passes. This is roughly

\[(\text{year number}) \cdot 4 + (\text{quartile number})\]

For example, the 3rd year, 2nd quartile: \(2 \cdot 4 + 2 = 10\)th quartile, so time_block = 10.

4.28 ScheduleSubjectRelation.__unicode__

Declaration  Declared in KroketApp/savemodels.py on line 273:
def __unicode__(self):

Qualified name  KroketApp.savemodels.ScheduleSubjectRelation.__unicode__

Description  The description of the schedule as String. Contains the schedule description and the subject description.

Parameters

  self – (ScheduleSubjectRelation) An instance of the ScheduleSubjectRelation.

Returns  (String) Description of the ScheduleSubjectRelation.

4.4.29  ScheduleSubjectRelation.Meta

Declaration  Declared in KroketApp/savemodels.py on line 278:
class Meta:

Qualified name  KroketApp.savemodels.ScheduleSubjectRelation.Meta

Description  A meta class which describes constraints and other meta-information.
CHAPTER 4. PYTHON (DJANGO)

4.5 KroketApp.urls

This file declares the URLs for the HTML and JavaScript files.

Hierarchical member index  This file contains the following members:

  urlpatterns ......................................................... 126

Detailed documentation for all declarations in this file follows.

4.5.1 urlpatterns

Declaration  Declared in KroketApp/urls.py on line 15:
urlpatterns = patterns('KroketApp',

Qualified name  KroketApp.urls.urlpatterns

Description  The URL patterns for Django to use.
   At the moment, this just transfers all calls to /query/ to KroketApp.query.urls. This could be extended to other URLs in the future, if necessary.
4.6 KROKETAPP.RECOMMODELS

4.6 KROKETAPP.recommodels

A Python script defining the database structure for recommending subjects. The data is fetched from the saved schedules of users in the database. Therefore it imports the models script of that database. → KROKETAPP.savemodules

Hierarchical member index  This file contains the following members:

<table>
<thead>
<tr>
<th>Member</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelatedSubject</td>
<td>127</td>
</tr>
<tr>
<td>subject_1</td>
<td>127</td>
</tr>
<tr>
<td>subject_2</td>
<td>127</td>
</tr>
<tr>
<td>rating</td>
<td>128</td>
</tr>
<tr>
<td>major</td>
<td>128</td>
</tr>
<tr>
<td>Meta</td>
<td>128</td>
</tr>
<tr>
<td>getRelatedSubjects</td>
<td>128</td>
</tr>
</tbody>
</table>

Detailed documentation for all declarations in this file follows.

4.6.1 RelatedSubject

Declaration  Declared in KROKETAPP/recommodels.py on line 14:
class RelatedSubject(models.Model):

Qualified name  KROKETAPP.recommodels.RelatedSubject

Description  A weak-entity for creating tuples between two subjects and a major, in which each tuple has a rating.

4.6.2 RelatedSubject.subject_1

Declaration  Declared in KROKETAPP/recommodels.py on line 18:
subject_1 = models.ForeignKey(ScheduleSubject, related_name='Subject_1')

Qualified name  KROKETAPP.recommodels.RelatedSubject.subject_1

Description  The first subject.

4.6.3 RelatedSubject.subject_2

Declaration  Declared in KROKETAPP/recommodels.py on line 22:
subject_2 = models.ForeignKey(ScheduleSubject, related_name='Subject_2')

Qualified name  KROKETAPP.recommodels.RelatedSubject.subject_2

Description  The second subject.
CHAPTER 4. PYTHON (DJANGO)

4.6.4 RelatedSubject.rating

Declaration  Declared in KroketApp/recommodels.py on line 27:

```
    rating = models.IntegerField()
```

Qualified name  KroketApp.recommodels.RelatedSubject.rating

Description  The rating on how much the combination of subject_1, subject_2 and major
occur in the schedules of the users. The higher the rating, the higher the frequency of this
tuple.

4.6.5 RelatedSubject.major

Declaration  Declared in KroketApp/recommodels.py on line 31:

```
    major = models.IntegerField()
```

Qualified name  KroketApp.recommodels.RelatedSubject.major

Description  The major of the schedule which has these two subjects.

4.6.6 RelatedSubject.Meta

Declaration  Declared in KroketApp/recommodels.py on line 37:

```
    class Meta:
```

Qualified name  KroketApp.recommodels.RelatedSubject.Meta

Description  Unique key is created by the two subjects and the major.

4.6.7 RelatedSubject.getRelatedSubjects

Declaration  Declared in KroketApp/recommodels.py on line 48:

```
    def getRelatedSubjects(subj):
```

Qualified name  KroketApp.recommodels.RelatedSubject.getRelatedSubjects

Description  Search for each tuple containing the given subject. If a tuple contains this
subject, then add this tuple to the result.

Parameters

```
    subj – (ScheduleSubject) The subject which will be used to find all related
    subjects.
```

Returns  (Queryset of ScheduleSubject) The result of the search.
4.7 KroketApp.security

A Python script that declares functions which will handle logging in, logging out, and checking integrity of an AJAX request. For example all queries should check their request parameter with check_validQuery.

The login data is found in request.session['loginInfo'] which contains a ‘username’ identifying the user and ‘schedule’ given the last loaded schedule.

→ KroketApp.query.user.authenticate
→ KroketApp.query.schedule.load

Hierarchical member index  This file contains the following members:

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Declared in KroketApp/security.py on line 20:</th>
</tr>
</thead>
<tbody>
<tr>
<td>SESSION_EXPIRING</td>
<td>SESSION_EXPIRING = 60*60</td>
</tr>
</tbody>
</table>

Qualified name   KroketApp.security.SESSION_EXPIRING

Description    The amount of seconds it takes before a session is expired.

4.7.1 SESSION_EXPIRING

4.7.2 check_credentials

Qualified name   KroketApp.security.check_credentials

Description    Checks whether the given credentials (username and password) are valid.

Parameters

name – (String) The username.
password – (String) The password.
CHAPTER 4. PYTHON (DJANGO)

Returns  (Boolean) Whether an account exists with the given username, and the password corresponds to this username.

4.7.3  check_validQuery

Declaration  Declared in KroketApp/security.py on line 49:
def check_validQuery(request, *arguments):

Qualified name  KroketApp.security.check_validQuery

Description  Checks whether the query is valid, i.e. if it is an AJAX POST request. If the request is not valid, a PermissionDenied is raised, else, nothing is done. Furthermore, this function also checks whether the required arguments are in given in the post request. Thus check_validQuery(request, 'arg1', 'arg2') checks if request is an AJAX request, a POST request and if 'arg1' and 'arg2' is given as parameter i.e. 'arg1' in request.POST; 'arg2' in request.POST. If these are not present a PermissionDenied is raised.

This will also update the session expiration date.

Parameters

request – (Request) The HTTP request object.
*arguments – (Var-amount of Strings) The arguments which should present in the request.

4.7.4  check_loggedIn

Declaration  Declared in KroketApp/security.py on line 65:
def check_loggedIn(request):

Qualified name  KroketApp.security.check_loggedIn

Description  A check that will make sure the user is logged in. If the user is logged in correctly the corresponding account is returned. If the user is not logged in correctly a PermissionDenied is raised.

Parameters

request – (Request) The HTTP request object.

Returns  (Account) The corresponding account.
4.7.5  check_loggedOut

**Declaration**  Declared in KroketApp/security.py on line 83:

```python
def check_loggedOut(request):
```

**Qualified name**  KroketApp.security.check_loggedOut

**Description**  A check that will make sure the user is logged out. If the user is logged in correctly a PermissionDenied is raised. First checks whether the logout request is valid. If the request is not valid, a PermissionDenied is raised, else, logs out the user.

**Parameters**

- `request` – *(Request)* The Http-request object.

4.7.6  commitSession

**Declaration**  Declared in KroketApp/security.py on line 96:

```python
def commitSession(request):
```

**Qualified name**  KroketApp.security.commitSession

**Description**  A method that will make sure all changes to the session of the current request are saved. Should only be called when modifying mutable objects in the session dictionary.

**Parameters**

- `request` – *(Request)* The HTTP request object.
Chapter 5

Python (queries) documentation

In this chapter, all of the Python code for the queries on the server side will be documented. Python declarations are ordered on the file they are in.

5.1 KroketApp.query.urls

This file declares the URLs for all of the queries.
Compare this to the JavaScript file urls.js, that does approximately the same thing on the client side.
This file contains no declarations.
5.2 KroketApp.query.major.list

A query-script which handles AJAX-requests to /studieplanner/query/major/list. This script will return a list of all majors.

Hierarchical member index This file contains the following members:

getlist ................................................................. 133

Detailed documentation for all declarations in this file follows.

5.2.1 getlist

Declaration Declared in KroketApp/query/major/list.py on line 19:
def getlist(request):

Qualified name KroketApp.query.major.list.getlist

Description An AJAX request handler for the list of majors function. A list of dictionaries (name, target_group) will be returned, where name is the name of the package and the target_group is the unique identifier.

Returns (List of dictionaries) A list of all majors as described above.
5.3  KroketApp.query.major.subjects

A query-script which handles AJAX-requests to /studieplanner/query/major/subjects. This script will return all subjects for the requested major in standard format.

→ subject.info.standardFormat

Hierarchical member index  This file contains the following members:

searchmajor .......................... 134

Detailed documentation for all declarations in this file follows.

5.3.1  searchmajor

Declaration  Declared in KroketApp/query/major/subjects.py on line 29:

```python
def searchmajor(request):
```

Qualified name  KroketApp.query.major.subjects.searchmajor

Description  An AJAX request handler for the subjects in major function. The method fetches its arguments from request.POST. It returns all mandatory subjects in the major in standard format → subject.info.standardFormat

Parameters

- **target_group**  *(String)* An integer encoded in a string identifying the major. The variable corresponds to target_group property given by major/list.

Returns  *(JSON list)* A JSON list containing all mandatory subjects in the major identified by target_group (as a dictionary).
5.4 KroketApp.query.package.search

A query script that handles AJAX requests to /studieplanner/query/package/search. This script will split the search string into words, stem them and query the database for packages with keywords. The top $x$ amount of matches will then be returned via JSON in standard format.

To achieve this, we use throughout this file the corpus of stemming of wordnet. For performance we do not use the NLTK string tokenizer as it is faster to search on more than intended using the database to analyze the sentence and pick keywords out the search string.

Currently the first request to this script has a load time of about three seconds, this is due to loading the wordnet corpus. However, as the script has been loaded in the database no noticeable latency is experienced. It would be a lesser improvement to automatically load the script on startup, as the public website should rarely reload its scripts.

Hierarchical member index  This file contains the following members:

- ALLOWED_TYPES
- VQSToDict
- fuzzySubjectSearch
- lmtzr
- searchpackage
- strip_accents

Detailed documentation for all declarations in this file follows.

5.4.1 ALLOWED_TYPES

Declaration  Declared in KroketApp/query/package/search.py on line 30:
ALLOWED_TYPES = [SubjectPacket.USE, SubjectPacket.COHERENT]

Qualified name  KroketApp.query.package.search.ALLOWED_TYPES

Description  Indicates on which type of SubjectPacket you are allowed to search. This can be SubjectPacket.USE or SubjectPacket.COHERENT. → KroketApp.models.SubjectPacket

5.4.2 lmtzr

Declaration  Declared in KroketApp/query/package/search.py on line 36:
lmtzr = WordNetLemmatizer()

Qualified name  KroketApp.query.package.search.lmtzr

Description  The global Lemmatizer, that is used to convert words into their basic form (e.g. ‘walking’ → ‘walk’). → nltk.stem.wordnet

135
CHAPTER 5. PYTHON (QUERIES)

5.4.3 strip_accents

Declaration  Declared in KroketApp/query/package/search.py on line 50:
def strip_accents(s):

Qualified name  KroketApp.query.package.search.strip_accents

Description  A helper method that strips all accents from a Unicode string.

If the given string is not Unicode, this method will return the original string again.

Parameters

s – (ASCII/Unicode string) The string to be stripped from accents.

Returns  (ASCII/Unicode string) The stripped string.

5.4.4 searchpackage

Declaration  Declared in KroketApp/query/package/search.py on line 87:
def searchpackage(request):

Qualified name  KroketApp.query.package.search.searchpackage

Description  An AJAX request handler for the ‘search on package’-function. The method
fetches its arguments from request.POST. All arguments are optional. Fuzzy search on the
searchTerm is done using fuzzySubjectSearch.

If an argument is not present in the request.POST dictionary the request handler will
act as if no constraints are given on that particular parameter, unless otherwise noted.

The output is given in JSON format. The output is given in standard format.

Parameters

searchTerm – (Optional, String) A search term to filter and order subjects by
searching the code, name, remarks, study_goal and content fields of the
Subject.objects database.

range – (Optional, String) A string in the format “start-end”, where start and end
are integers, with

start < end and end - start < 100.

This indicates which part of the result is viewed. The bounds are inclusive and
starting at 0. For example, range="0-99" gives the top 100 results of the search
query. Default value is 0-19. If the string is not in the correct format a 403
Forbidden error will be given. If the integers are not in the correct range the
default value will be used.
5.4. KROKETAPP.QUERY.PACKAGE.SEARCH

type – (Optional, Integer) An indication in which type of packets is to be searched. Can be either SubjectPacket.USE or SubjectPacket.COHERENT. → KroketApp.models.SubjectPacket Any other value than those two will set the restriction to the default: both SubjectPacket.USE and SubjectPacket.COHERENT.

Returns  (JSON list) A JSON list of dictionaries with the fields: name, remarks, target_group and type. This describes the packages found. They are ordered to from best-match to worst-match.

5.4.5  fuzzySubjectSearch

Declaration  Declared in KroketApp/query/package/search.py on line 135:
def fuzzySubjectSearch(query, constraints, start, end):

Qualified name  KroketApp.query.package.search.fuzzySubjectSearch

Description  A method implementing the ‘fuzzy subject search’. The query string will first be stripped of any accents. (For instance é will be replaced with e). All symbols will be removed and the query is split into words. All possible stems are then found using wordnet. Using these stems the KeywordPacketRelation will be searched for all keywords which contains one of the entered keywords. The KeywordPacketRelation will then be grouped per subject and the sum of the counts per subject will be calculated. A descending ordering on this sum will be applied and the slice starting at index start and ending at index end will be taken.

Parameters

query – (String) The searchTerm that should be used to search the database for subjects.
constraints – (Q) A Q object that will be applied prior on KeywordPacketRelation prior searching.
start – (Integer) An integer indicating the starting index of the slice requested.
end – (Integer) An integer indicating the end index of the slice requested.

Returns  (ValueQuerySet) A ValueQuerySet of the KeywordPacketRelation with the following fields: target_group, target_group_name, target_group_remarks, target_group_type and rating (which is the sum of all counts in the matching KeywordPacketRelation, the higher the number, the better the match is). It will be sorted descending on the rating.
CHAPTER 5. PYTHON (QUERIES)

5.4.6 VQSToDict

Declaration  Declared in KroketApp/query/package/search.py on line 162:

```python
def VQSToDict(vqs):
```

Qualified name  KroketApp.query.package.search.VQSToDict

Description  An internal method used only in this script. Converts a ValueQuerySet from KeywordPacketRelation to a list of dictionaries in with fields: name, remarks, target_group and type. This ValueQuerySet is related to a list of dictionaries (although not the same, and cannot be converted using simpleJSON).

Parameters

- `vqs` – (ValueQuerySet) A value query set of the KeywordPacketRelation with the following fields: target_group_name, target_group_remarks, target_group and target_group_type.

Returns  (JSON list) A JSON list of dictionaries describing the target_groups.
5.5 KroketApp.query.package.subjects

A query script that handles AJAX requests to /studieplanner/query/package/subjects. This script will return all subjects for the request package in standard format.

Hierarchical member index  This file contains the following members:

    searchpackage .................................................... 139

Detailed documentation for all declarations in this file follows.

5.5.1 searchpackage

Declaration  Declared in KroketApp/query/package/subjects.py on line 22:

    def searchpackage(request):

Qualified name  KroketApp.query.package.subjects.searchpackage

Description  An AJAX request handler for the subjects in package function The method fetches its arguments from request.POST. The returned subjects are in standard format → subject.info.standardFormat.

Parameters

    target_group (String) An integer encoded in a string identifying the package.
    The variable corresponds to target_group property given by package/search.

Returns  (JSON list) A JSON list containing all subjects in the package identified by target_group (as a dictionary)
5.6 KroketApp.query.recommendation.subject

A query-script which handles AJAX-requests to /studieplanner/query/recommendation/subject. The script will try to find related subject

Hierarchical member index  This file contains the following members:

- AMOUNT_OF_RESULTS
- view

Detailed documentation for all declarations in this file follows.

5.6.1 AMOUNT_OF_RESULTS

Declaration  Declared in KroketApp/query/recommendation/subject.py on line 17:

```
AMOUNT_OF_RESULTS = 8
```

Qualified name  KroketApp.query.recommendation.subject.AMOUNT_OF_RESULTS

Description  The maximal amount of results the AJAX-query should return.

5.6.2 view

Declaration  Declared in KroketApp/query/recommendation/subject.py on line 27:

```
def view(request):
```

Qualified name  KroketApp.query.recommendation.subject.view

Description  Given a list of chosen subject (in minimal form), this query will reply with a list of recommended subjects. The replied subjects are in standard format as defined in query.subject.info. → subject.info.standardFormat

Parameters

- subjects – (JSON-List) A JSON list with all non-mandatory subject-codes in the schedule.
- major – (Integer) The target_group identifier of the major. If no major is chosen this should be -1.

Returns  (JSON-list) A JSON list with the subject-codes of the recommended subjects.
5.7 KroketApp.query.schedule.load

A query-script which handles AJAX-requests to /studieplanner/query/user/load. This script loads the schedule of a user from the database and returns (if it exists) the schedule of that user. → KroketApp.savemodels

Hierarchical member index  This file contains the following members:

- emptyResponse ........................................ 143
- generateResponse .................................... 142
- getMajorName ........................................ 143
- getSchedule .......................................... 142
- loadschedule ......................................... 141
- translateQStoJSON .................................... 143

Detailed documentation for all declarations in this file follows.

5.7.1 loadschedule

Declaration  Declared in KroketApp/query/schedule/load.py on line 58:

def loadschedule(request):

Qualified name  KroketApp.query.schedule.load.loadschedule

Description  Main method that is responsible for loading the schedule of a user. The input relies solely on the database. If the user has no session on the database, then no permission will be granted. The output is a dictionary which is formatted in JSON.

Argument in the dictionary of the input is:

- **username**: string, the username of the account of the user.

- **savename**: string, the name of the schedule.

Arguments in the dictionary of the output are:

- **schedule**: list of list of strings, which represent the time blocks and subject codes. For more explanation see method translateQStoJSON.

- **major**: integer, the primary key for the major saved with the schedule (-1 if none).

- **majorName**: string, the name of the major.

- **year**: integer, the year in which the first quartile starts.

- **success**: boolean, True if the user has a schedule on the database, i.e. has saved a schedule with this account before.
CHAPTER 5. PYTHON (QUERIES)

Parameters

request – *(request)* Must have the username of the account of the user in the
session to perform this action, i.e. user must be logged in and a name for a
schedule must be given. If this name matches to a name of a schedule in the
database, return that schedule, otherwise return dummy schedule.

Returns  *(JSON)* The user’s schedule formatted in JSON.

5.7.2 getSchedule

Declaration  Declared in KroketApp/query/schedule/load.py on line 76:
def getSchedule(account, schedule_name):

Qualified name  KroketApp.query.schedule.load.getSchedule

Description  Retrieves the user’s schedule from the database. If it does not exist, the
method will raise an exception.

Parameters

dbAccount – *(Unicode string)* The username of the user’s account.

Returns  *(Dictionary)* A dictionary containing schedule, containing various values related
to the schedule, and ssr, the list of lists with Subjects.

5.7.3 generateResponse

Declaration  Declared in KroketApp/query/schedule/load.py on line 102:
def generateResponse(listSsr, schedule):

Qualified name  KroketApp.query.schedule.load.generateResponse

Description  Formats the result of the main method in the correct method.

Parameters

listSsr – *(List of list of strings)* Represents the user’s schedule.
schedule – *(Schedule)* Containing values relevant to the schedule: ID, major,
begin year, account related to.

Returns  *(Dictionary)* A dictionary with all values relevant to the schedule.
5.7.4 emptyResponse

Declaration  Declared in KroketApp/query/schedule/load.py on line 116:

def emptyResponse():

Qualified name  KroketApp.query.schedule.load.emptyResponse

Description  When the user’s schedule does not exist on the database, still initialize dictionary with non-usable data

Returns  (Dictionary) Dictionary with non-usable data and false to indicate a non-existing schedule on the database.

5.7.5 translateQStoJSON

Declaration  Declared in KroketApp/query/schedule/load.py on line 133:

def translateQStoJSON(dbSchedule):

Qualified name  KroketApp.query.schedule.load.translateQStoJSON

Description  When the schedule is retrieved from the database, it is formatted in QuerySet. It needs to be translated to a list of list of Subjects. However, time block i is stored as integer in the QuerySet, but will be translated to the i\textsuperscript{th} position in the list. Also in the QuerySet only the code of the subject is stored, while other values related to that subject also needs to be retrieved.

Parameters

\[\text{dbSchedule} \rightarrow (\text{QuerySet}) \text{ The user’s schedule from the database.}\]

Returns  (List of list of Subject) The user’s schedule formatted.

5.7.6 getMajorName

Declaration  Declared in KroketApp/query/schedule/load.py on line 156:

def getMajorName(major):

Qualified name  KroketApp.query.schedule.load.getMajorName

Description  Retrieves the name of the major.
CHAPTER 5. PYTHON (QUERIES)

Parameters

major – (integer) The primary key of the major.

Returns  (string) The name of that major.
5.8 KroketApp.query.schedule.save

A query-script which handles AJAX-requests to /studieplanner/query/user/save. This script saves the schedule of a user to the database → KroketApp.savemodels.

Hierarchical member index  This file contains the following members:

- checkScheduleOverwrite
- decodeJSON
- isCorrectName
- returnSubject
- saveScheduleToDb
- saveschedule

Detailed documentation for all declarations in this file follows.

5.8.1 saveschedule

Declaration  Declared in KroketApp/query/schedule/save.py on line 47:

def saveschedule(request):

Qualified name  KroketApp.query.schedule.save.saveschedule

Description  Main method that is responsible for saving the schedule of a user. All arguments needed to complete this request should be in the request.POST, which is also the only input. However, if the user is not logged in, i.e. the username of the account is not in a session, then permission denied is raised. If no exceptions are raised, then the schedule is saved.

Arguments in the dictionary of the input are:

- **username**: string, the username of the account of the user in the session.
- **schedule**: list of list of strings, which represent the time blocks and subject codes. For more explanation see method translateQStoJSON in /query/user/load.py.
- **major**: integer, the primary key for the major saved with the schedule (-1 if none).
- **year**: integer, the year in which the first quartile starts.
- **savename**: string, the name of the schedule

Parameters

- **request** – (request) Contains a dictionary with the values necessary to save the user's schedule.
CHAPTER 5. PYTHON (QUERIES)

Returns (String) An empty string if successful. (If the saving was not successful, a PermissionDenied is issued.

5.8.2 isCorrectName

Declaration Declared in KroketApp/query/schedule/save.py on line 73:
def isCorrectName(name):

Qualified name KroketApp.query.schedule.save.isCorrectName

Description Checks whether the name of the schedule is allowed.

Parameters

username – (String) The name of the schedule, which should be checked.

Returns (Boolean) true if

- the name only contains letters (A-Z or a-z), numbers (0-9), dashes (-), underscores (_), and periods (,);
- the name contains no more than one period in a row;
- the length of the name is inside the range [1,80].

Otherwise false is returned.

5.8.3 returnSubject

Declaration Declared in KroketApp/query/schedule/save.py on line 82:
def returnSubject(subj_code):

Qualified name KroketApp.query.schedule.save.returnSubject

Description Returns the ScheduleSubject providing the code. Also adds the subject to the database if it does not exist yet.

Parameters

subj_code – (String) The code of the subject.

Returns (ScheduleSubject) The ScheduleSubject of the given code.
5.8. KROKETAPP.QUERY.SCHEDULE.SAVE

5.8.4 checkScheduleOverwrite

Declaration  Declared in KroketApp/query/schedule/save.py on line 97:
def checkScheduleOverwrite(account, schedule_name):

Qualified name  KroketApp.query.schedule.save.checkScheduleOverwrite

Description  If a schedule with the same name already exists, then delete that schedule.

Parameters
   account – (Account) The account of the user.
   schedule_name – (String) The name of the schedule.

5.8.5 saveScheduleToDb

Declaration  Declared in KroketApp/query/schedule/save.py on line 114:
def saveScheduleToDb(schedule, account, major, begin_year, schedule_name):

Qualified name  KroketApp.query.schedule.save.saveScheduleToDb

Description  Saves a schedule to the database.

Parameters
   schedule – (List of list of string) The user’s schedule.
   account_name – (String) The account name of the user’s account.
   major – (Integer) The major of the schedule.
   begin_year – (Integer) The begin year of the schedule.
   schedule_name – (String) The name of the schedule.

5.8.6 decodeJSON

Declaration  Declared in KroketApp/query/schedule/save.py on line 143:
def decodeJSON(json_schedule):

Qualified name  KroketApp.query.schedule.save.decodeJSON

Description  Decodes JSON format to a QuerySet.

Parameters
   json_schedule – (JSON) The schedule, list of list of Subject, in JSON format.

Returns  (list of list of String) The schedule formatted in list of list of Subject.
CHAPTER 5. PYTHON (QUERIES)

5.9 KroketApp.query.schedule.list

A query-script which handles AJAX-requests to /studieplanner/query/schedule/list. This script will return all saved schedules of the user logged in in the current session.

Hierarchical member index  This file contains the following members:

userschedules ........................................... 148

Detailed documentation for all declarations in this file follows.

5.9.1 userschedules

Declaration  Declared in KroketApp/query/schedule/list.py on line 18:
def userschedules(request):

Qualified name  KroketApp.query.schedule.list.userschedules

Description  An AJAX request handler for the ‘list of schedules’ function. A list of dictionaries (name) will be returned. Here is name the name of the schedule. There are no parameters. It is required that a user is logged in in the current session.

Returns  (List of dictionaries) A list of all schedules of the current user.
5.10. KroketApp.query.schedule.rename

A query-script which handles AJAX-requests to /studieplanner/query/schedule/rename. This script will implement a query to change the name of a schedule. The user must be logged in to be able to use this function.

Hierarchical member index  This file contains the following members:

- FALSE ................................. 149
- TRUE ................................. 149
- renameScheduleName .................. 149

Detailed documentation for all declarations in this file follows.

5.10.1  FALSE

Declaration  Declared in KroketApp/query/schedule/rename.py on line 16:
FALSE = False

Qualified name  KroketApp.query.schedule.rename.FALSE

Description  A constant representing a false result value

5.10.2  TRUE

Declaration  Declared in KroketApp/query/schedule/rename.py on line 21:
TRUE = True

Qualified name  KroketApp.query.schedule.rename.TRUE

Description  A constant representing a true result value

5.10.3  renameScheduleName

Declaration  Declared in KroketApp/query/schedule/rename.py on line 36:
def renameScheduleName(request):

Qualified name  KroketApp.query.schedule.rename.renameScheduleName

Description  An AJAX request handler for the ‘rename a schedule’ function. A dictionary will be returned with the key ‘success’ which will indicate if the schedule was renamed correctly or if something went wrong. This could be because the schedule with the given name didn’t exist or a schedule with the new name already existed.
CHAPTER 5. PYTHON (QUERIES)

Parameters

- **oldName** – *(String)* The current/old name of the schedule which should be renamed.
- **newName** – *(String)* The new name of the schedule.

Returns *(JSON-dictionary)* A dictionary with the key ‘success’ with value TRUE or FALSE indicating if the renaming occurred or something went wrong, respectively.
5.11. KROKETAPP.QUERY.SCHEDULE.DELETE

5.11 KroketApp.query.schedule.delete

A query-script which handles AJAX-requests to /studieplanner/query/schedule/delete. This script will remove a certain schedule. The user has to be logged in the current session to have this script enabled.

Hierarchical member index  This file contains the following members:

- deleteSchedule .......................................... 151
- removeSchedule .......................................... 151

Detailed documentation for all declarations in this file follows.

5.11.1 removeSchedule

Declaration  Declared in KroketApp/query/schedule/delete.py on line 20:
def removeSchedule(request):

Qualified name  KroketApp.query.schedule.delete.removeSchedule

Description  An AJAX request handler for the ‘delete a schedule’ function. Parameters are fetched from request.POST Given the parameter name, the function will try to delete a schedule belonging to the user with the given name. If this is done properly dictionary with key ‘success’ and value ‘True’, if no schedule could be found the value will be ‘False’.

Parameters

- name – (String) The name of the schedule.

Returns  (JSON-dictionary) A dictionary with key ‘success’ indicating if the schedule was deleted.

5.11.2 deleteSchedule

Declaration  Declared in KroketApp/query/schedule/delete.py on line 36:
def deleteSchedule(dbSchedule):

Qualified name  KroketApp.query.schedule.delete.deleteSchedule

Description  Deletes a schedule and all its relations.

Parameters

- dbSchedule – (Schedule) The schedule which should be deleted from the database.
CHAPTER 5. PYTHON (QUERIES)

5.12 KroketApp.query.schedule.validate

A query-script which handles AJAX-requests to /studieplanner/query/schedule/validate. This query will analyze a schedule, which consists of a list of subjects and a chosen major, and will give an indication if the exam-committee will accept this planning straight away, or will have to discuss about the planning.

The validation is done in separate parts:

- Check if all mandatory major subjects are scheduled. This is done in validate.
- Check if there is at least one USE package scheduled. This is done in checkUSE.
- Check if there is at least two coherent package scheduled. This is done in checkCoherent.
- Check if the USE and coherent Packages have no subject in common. This is done in checkOverlap.

This implementation of validation replaces the old validation.js since svn revision ±5000. We have moved the validation checks to the server to avoid big data-traffic from server to the client, as the validation is data-intensive. It also avoids giving away too much information about our database, avoiding CSRF. As validation.js depended on the AJAX-queries: /studieplanner/query/electivepackage/list and /studieplanner/query/usepackage/list, and these are only implemented to aid the old validation check, this script deprecates the two AJAX-queries.

The implementation does not handle the case when there are non-mandatory subjects in a package. To implement this correctly, the GUI would also have to be able to distinguish them (using colours).

Hierarchical member index  This file contains the following members:

- MIN_COHERENT ........................................ 152
- MIN_USE .................................................... 153
- checkCoherent ........................................... 155
- checkOverlap ............................................ 155
- checkUSE .................................................. 154
- inPackage ............................................... 156
- validate ................................................ 153

Detailed documentation for all declarations in this file follows.

5.12.1 MIN_COHERENT

Declaration  Declared in KroketApp/query/schedule/validate.py on line 35:
MIN_COHERENT = 2

Qualified name  KroketApp.query.schedule.validate.MIN_COHERENT
5.12. KROKETAPP.QUERY.SCHEDULE.VALIDATE

**Description**  The amount of coherent packages there should be minimal in a schedule to have the package automatically approved by the exam-committee.

### 5.12.2 MIN_USE

**Declaration**  Declared in KroketApp/query/schedule/validate.py on line 41:

```python
MIN_USE = 1
```

**Qualified name**  KroketApp.query.schedule.validate.MIN_USE

**Description**  The amount of use packages there should be minimal in a schedule to have the package automatically approved by the exam-committee.

### 5.12.3 validate

**Declaration**  Declared in KroketApp/query/schedule/validate.py on line 96:

```python
def validate(request):
```

**Qualified name**  KroketApp.query.schedule.validate.validate

**Description**  An AJAX request handler for the validate a schedule function The method fetches its arguments from request.POST. Given all subject in the schedule and the major, this query will response with a JSON-object with the following fields:

- **‘use’** : Information about the USE-packages in the schedule:
  - **‘status’** : ‘PASS’ if there is at least MIN_USE USE packages in the schedule, otherwise ‘FAIL’
  - **‘packages’** : A list of names of all USE packages contained in the schedule; the subjects in each package can overlap with subjects in the other USE/elective packages.

- **‘coherent’** : Information about the coherent-packages in the schedule:
  - **‘status’** : ‘PASS’ if there is at least MIN_COHERENT elective packages in the schedule, otherwise ‘FAIL’
  - **‘packages’** : A list of names of all coherent-packages, contained in the schedule; the subjects in each package can overlap with subjects in the other USE/elective packages.

- **‘major’** : Information about the subjects of the major in the schedule
  - **‘status’** : ‘PASS’ if all mandatory subjects of the major scheduled in the schedule, otherwise ‘FAIL’.
  - **‘missing’** : A list of subject-codes of the subjects which are mandatory subjects of the given major, but are not planned in the given schedule.
CHAPTER 5. PYTHON (QUERIES)

- ‘overlap’: Information about packages in the schedule, which have subjects in common with other packages in the schedule:
  - ‘status’: ‘PASS’ if no USE- or coherent-package has a subject in common with a USE- or coherent-package (not necessarily of the same type as the first package), otherwise ‘FAIL’.
  - ‘overlap’: Information about the subjects which are in two or more packages (where the packages must be completely in the schedule), is a list:
    * ‘code’: The subject code of the package that is in two or more packages.
    * ‘inPackage’: A list of package names in which the subject is contained.

Note that a status ‘PASS’ in all four fields does not guarantee that the exam committee will approve the schedule, it is rather a guideline.


Parameters

schedule – (JSON list) A JSON list of subject-codes that are in the schedule.
major – (Integer) A target_group identifier of the chosen major, if no major exists corresponding to the major the result of this query is undefined (though no internal data will be changed).

Returns (JSON-object) A JSON object with information about the USE packages, elective packages, major and overlapping packages with respect to the schedule, as described above.

5.12.4 checkUSE

Declaration Declared in KroketApp/query/schedule/validate.py on line 132:
def checkUSE(subjects):

Qualified name KroketApp.query.schedule.validate.checkUSE

Description An internal method which will find all USE-packages that have all subjects in the given subjects parameter. It will return a tuple with as first element the set of target_group identifiers of the USE-packages (used to check for overlapping packages) and in the last element a dictionary with the following structure:

- ‘status’: ‘PASS’ if there is at least MIN_USE USE-packages in the schedule, otherwise ‘FAIL’
- ‘packages’: A list of names of all use-packages contained in the schedule, the subjects in each package can overlap with subjects in the other USE/coherent-packages.
5.12. KROKETAPP.QUERY.SCHEDULE.VALIDATE

Parameters

subjects – (Set-of-Subject) The set of subject in which we want to find the USE-packages.

Returns (Set-of-Integers, Dictionary) A set of target_group identifiers with the USE-packages and a data-structure to be returned to the client

5.12.5 checkCoherent

Declaration Declared in KroketApp/query/schedule/validate.py on line 165:
def checkCoherent(subjects):

Qualified name KroketApp.query.schedule.validate.checkCoherent

Description An internal method which will find all coherent-packages that have all subjects in the given subjects parameter. It will return a tuple with as first element the set of target_group identifiers of the coherent-packages (used to check for overlapping packages) and in the last element a dictionary with the following structure:

- ‘status’: ‘PASS’ if there is at least MIN_COHERENT USE-packages in the schedule, otherwise ‘FAIL’
- ‘packages’: A list of names of all use-packages contained in the schedule, the subjects in each package can overlap with subjects in the other USE/coherent-packages.

Parameters

subjects – (Set-of-Subject) The set of subject in which we want to find the coherent-packages.

Returns (Set-of-Integers, Dictionary) A set of target_group identifiers with the coherent-packages and a data-structure to be returned to the client

5.12.6 checkOverlap

Declaration Declared in KroketApp/query/schedule/validate.py on line 200:
def checkOverlap(packages):

Qualified name KroketApp.query.schedule.validate.checkOverlap
CHAPTER 5. PYTHON (QUERIES)

Description  An internal method which will find all subjects which are in more than one package of the package set packages. It will return a dictionary with the following structure:

- ‘status’: ‘PASS’ if no USE- or coherent-package has a subject in common with a USE- or coherent-package (not necessarily of the same type as the first package), otherwise ‘FAIL’.
- ‘overlap’: Information about the subjects which are in two or more packages (where the packages must be completely in the schedule):
  - ‘subject’: The subject code of the package that is in two or more packages.
  - ‘inPackage’: A list of package-names, in which the subject is contained.

Parameters

packages – (Set-of-Integers) A set of target group identifiers, identifying the given packages.

Returns  (Dictionary) A dictionary describing the overlapping packages as defined above.

5.12.7 inPackage

Declaration  Declared in KroketApp/query/schedule/validate.py on line 226:
def inPackage(subject,packages):

Qualified name  KroketApp.query.schedule.validate.inPackage

Description  A helper method which find in which packages a given subject is scheduled, given a set of packages.

Parameters

subject – (String) The subject code of the package.
packages – (Set-of-Integers) A set of target group identifiers, identifying the given packages.

Returns  (List-of-Strings) A list of names of the packages, inside packages, in which the scheduled is planned
5.13. KroketApp.query.subject.info

A query-script which handles AJAX-requests to /studieplanner/query/subject/info. This script will return for a given subject-code all information available on the server about the subject. This is done in the extended-format of subject. In this script some helper functions are also defined. The helper-functions defined are:

- getPlanning of subject
- prior_knowledge of subject
- standardFormat representation, which is the default return method of a subject for queries
- extendedFormat representation, which contains all information available about a subject

Hierarchical member index  This file contains the following members:

- convert .................................................. 158
- extendedFormat ...................................... 161
- follow_up .............................................. 160
- getPlanning ............................................ 159
- info ...................................................... 157
- prior_knowledge ................................. 159
- standardFormat ................................. 160
- toStandardFormat ......................... 158

Detailed documentation for all declarations in this file follows.

5.13.1  info

Declaration  Declared in KroketApp/query/subject/info.py on line 33:

```python
def info(request):
```

Qualified name  KroketApp.query.subject.info.info

Description  An AJAX request handler, for the subject information function The method fetches its arguments from request.POST. Given a subject-code this method fetches all available information and returns this as a JSON-dictionary in the same format as defined in extended-format. In case the subject does not exists a 403 PermissionDenied is raised. → extended-format

Parameters

- code – (String) The subject-code from which the information should be fetched.
CHAPTER 5. PYTHON (QUERIES)

Returns (JSON-Dictionary) Information about the subject in extended format.

5.13.2 convert

Declaration  Declared in KroketApp/query/subject/info.py on line 57:
def convert(request):

Qualified name  KroketApp.query.subject.info.convert

Description  Converts an arbitrary deep nested list of subject-code into a nested list of subject in standard format. That is given a list or String this function p will behave as follow:

- If a list is given, all elements of this list will be converted using p’, and removed if the result is none.
- If a subject-code is given, a subject with matching subject-code is found and converted. But if a matching subject can not be found None is given.

Now p is returned. → toStandardFormat

Parameters

  subjects – (JSON-List) An arbitrary deep nested list of subject codes.

Returns  (JSON-List) The list in the same order but with all correct subject-code replaced by the corresponding subject as dictionary in standardFormat. Incorrect subject-codes will be removed.

5.13.3 toStandardFormat

Declaration  Declared in KroketApp/query/subject/info.py on line 78:
def toStandardFormat(data):

Qualified name  KroketApp.query.subject.info.toStandardFormat

Description  Converts an arbitrary deep nested list of subject-code into a nested list of subject in standard format. That is given a list or String this function p will behave as follow:

- If a list is given, all elements of this list will be converted using p’, and removed if the result is none.
- If a subject-code is given, a subject with matching subject-code is found and converted. But if a matching subject can not be found None is given.

Now p is returned.
Parameters

data – (List/String) An arbitrary deep nested list of subject codes.

Returns  (List/StandardFormat-Subject) The list in the same order but with all correct subject-code replaced by the corresponding subject as dictionary in standardFormat. Incorrect subject-codes will be removed.

5.13.4  getPlanning

Declaration  Declared in KroketApp/query/subject/info.py on line 107:
def getPlanning(pk):

Qualified name  KroketApp.query.subject.info.getPlanning

Description  A function to get all plannings from a subject as a list of dictionaries. Used for the standard and extended format. In the dictionaries the following keys are present:

- year: The year in which the subject is given (an integer 1-3);
- quartile: The corresponding quartile of the year in which the subject is given;
- timeSlot: The corresponding timeslot (A-E), separated by semicolons (’;’), if the subject has lectures/instructions on multiple timeslots;
- packet: The packet (major/USE-/Elective-package) for which the planning is meant for, is given as name;
- packetType: The packetType: SubjectPacket.USE, SubjectPacket.MAJOR, SubjectPacket.COHERENT;
- mandatory: Whether the subject is mandatory when the packetType is SubjectPacket.MAJOR.

Parameters

pk – (String) The primary key/the subject-code of the subject, from which the planning should fetched.

Returns  (List-of-dictionaries) A list of dictionaries with in each dictionary information about when and for which packet the subject is planned.

5.13.5  prior_knowledge

Declaration  Declared in KroketApp/query/subject/info.py on line 125:
def prior_knowledge(pk):
CHAPTER 5. PYTHON (QUERIES)

Qualified name KroketApp.query.subject.info.prior_knowledge

Description A function to get all prior knowledge required from a subject as a string Used for the standard and extended format.

Parameters

pk – (String) The primary key/the subject-code of the subject, from which the prior knowledge should fetched.

Returns (String) A string with all subject which are prior knowledge to this subject, multiple subjects are concatenated with the semicolon character (’;’).

5.13.6 follow_up

Declaration Declared in KroketApp/query/subject/info.py on line 140:
def follow_up(pk):

Qualified name KroketApp.query.subject.info.follow_up

Description A function to get all subjects which are follow up subjects to this subject as a string Used for the extended format.

Parameters

pk – (String) The primary key/the subject-code of the subject, from which the follow up subjects should fetched.

Returns (String) A string with all subject which are prior knowledge to this subject, multiple subjects are concatenated with the semicolon character (’;’).

5.13.7 standardFormat

Declaration Declared in KroketApp/query/subject/info.py on line 168:
def standardFormat(subject):

Qualified name KroketApp.query.subject.info.standardFormat
**Description**  Given an instance of a Subject model, this method fetches information known about the subject in the database. All queries should use this function to convert their subject into standardFormat to be returned for queries. However, for performance reasons exceptions may exist. Currently the only exception is in the subject.search script → subject.search.VQSToDict. It is returned as a dictionary with the following keys:

- **code**: The subject code of the subject
- **name**: The name of the subject
- **ects**: The amount of ects this subject is worth
- **difficulty**: The niveau of the subject, which can be (here represented as String) Basic, Intermediate and Advanced
- **lastYear**: The lastYear in which the subject is given, if this is unknown the value is 9999
- **planning**: A list of dictionaries describing how and when the colleges of the subjects are given → `getPlanning`
- **priorKnowledge**: A string with all subject which are prior knowledge to this subject, multiple subjects are concatenated with the semicolon character (‘;’). → `priorKnowledge`

**Parameters**

subject – (Subject) The subject from which the standard format should be generated.

**Returns**  (Dictionary) A dictionary of the subject in standard format.

### 5.13.8 extendedFormat

**Declaration**  Declared in KroketApp/query/subject/info.py on line 206:

def extendedFormat(subject):

**Qualified name**  KroketApp.query.subject.info.extendedFormat

**Description**  Given an instance of a Subject model, this method fetches all information known about the subject in the database. It is returned as a dictionary with the following keys:

- **code**: The subject code of the subject
- **name**: The name of the subject
- **ects**: The amount of ects this subject is worth
CHAPTER 5. PYTHON (QUERIES)

- **difficulty**: The niveau of the subject, which can be (here represented as String) basic, intermediate and advanced.

- **lastYear**: The lastYear in which the subject is given, if this is unknown the value is 9999

- **remarks**: The remarks of the subject

- **studyGoal**: The study goal of the subject

- **content**: The content description of the subject

- **weeklyContent**: The weekly content description of the subject

- **videoUrl**: The URL to the video-college site of the subject

- **videoMaterialDescription**: The description of the video material

- **educationType**: The education type of the subject

- **examinationType**: The examination type used by the subject

- **department**: The department that is responsible for the subject

- **subdepartment**: The subdepartment that is responsible for the subject

- **planning**: A list of dictionaries describing how and when the colleges of the subjects are given → getPlanning

- **priorKnowledge**: A string with all subject which are prior knowledge to this subject, multiple subjects are concatenated with the semicolon character (‘;’). → prior_knowledge

- **followUp**: A string with all subject which are follow up to this subject, multiple subjects are concatenated with the semicolon character (‘;’). → follow_up

**Parameters**

subject – *(Subject)* The subject from which the extended format should be generated.

**Returns** *(Dictionary)* A dictionary of the subject in extended format.
5.14 KROKETAPP.QUERY.SUBJECT.SEARCH

5.14 KroketApp.query.subject.search

A query-script which handles AJAX-requests to /studieplanner/query/subject/search. This script will split search string into words, stem them and query the database for subjects with keywords. The top x amount of matches will then be returned via JSON in standard-format.

To achieve this, we use throughout this file the corpus of stemming of wordnet. For performance we do NOT use the nltk string tokenizer as it is faster to search on more then intended using the database than to analyze the sentence and pick keywords out the search string.

Currently the first request to this script has a load time of 3+ seconds, this is due to loading the wordnet corpus. However, as the script has been loaded in the database no noticeable latency is found. It would be a lesser improvement to automatically load the script on startup, as the public website should rarely reload its scripts.

Hierarchical member index  This file contains the following members:

- VQSToDict ................................................................. 166
- fuzzySubjectSearch ..................................................... 165
- lmtzr ................................................................. 163
- strip_accents ............................................................ 163
- subject ............................................................... 164

Detailed documentation for all declarations in this file follows.

5.14.1 lmtzr

Declaration  Declared in KroketApp/query/subject/search.py on line 32:
lmtzr = WordNetLemmatizer()

Qualified name  KroketApp.query.subject.search.lmtzr

Description  The global Lemmatizer, that is used to convert words into their basic form (e.g. ‘walking’ → ‘walk’). → nltk.stem.wordnet.

5.14.2 strip_accents

Declaration  Declared in KroketApp/query/subject/search.py on line 46:
def strip_accents(s):

Qualified name  KroketApp.query.subject.search.strip_accents

Description  A helper method that strips all accents from a Unicode string.
If the given string is not Unicode, this method will return the original string again.
5. PYTHON (QUERIES)

Parameters

\( s \) – (ASCII/Unicode string) The string to be stripped from accents.

Returns  (ASCII/Unicode string) The stripped string.

5.14.3 subject

Declaration  Declared in KroketApp/query/subject/search.py on line 95:
def subject(request):

Qualified name  KroketApp.query.subject.search.subject

Description  An AJAX request handler for the ‘search on subject’-function. The method fetches its arguments from request.POST. All arguments are optional. FuzzySearch on searchTerm is done using \( \rightarrow \) fuzzySubjectSearch.

If an argument is not present in the request.POST dictionary the request handler will act as if no constraints are given on that particular subject, unless otherwise noted.

The output is given in JSON format. The output is given in standard format.

Parameters

range – (Optional, String) A string in the format "start-end", where start and end are integers, with

\[
\text{start} < \text{end} \quad \text{and} \quad \text{end} - \text{start} < 100.
\]

This indicates which part of the result is viewed. The bounds are inclusive and starting at 0. For example, range="0-99" gives the top 100 results of the search query. Default value is 0~19. If the string is not in the correct format a 403 Forbidden error will be given. If the bounds are not fulfilled a permission denied will be thrown.

searchTerm – (Optional, String) a searchTerm which will filter and order subjects; by searching the code, name, remarks, study_goal and content fields of the Subject.objects database.

timeSlot – (Optional, String) The possible time slot in which the subject can be given, possible values ‘A’-’E’. If a subject is given (can be more, this subject will be found)

quartile – (Optional, String) This represents the quartile in which the subject may be given. A quartile is represented by an integer between 1 and 4 (inclusive).

year – (Optional, String) The academic year in which you normally take the course as string which is a number between 1 and 3 format.
difficulty – (Optional, String) The difficulty is represented by an integer between 1 and 3 (inclusive). For instance, a difficulty of 3 indicates that the subject is normally taken in the 3rd year of the bachelor course.

target_group – (Optional, Integer) An integer identifying the target group of which the subject should be part of.

major – (Optional, String) Indicates in which the user is majoring in. The string is a single code as described in: http://w3.tue.nl/nl/onderwijs/tue_bachelor_college/studieopbouw/major/.

broadening – (Optional, String) A character being either “y” (yes) or “n” (no). This indicates if the course should be broadening in respect to the major field (thus not from the same study) or (in the case of “n”) should be deepening. If they can be both the field should be empty. It should be noted that the major argument should be present.

Returns  (JSON-list) A JSON-list of subjects in standardFormat. → info.standardFormat.

5.14.4 fuzzySubjectSearch

Declaration  Declared in KroketApp/query/subject/search.py on line 174:

def fuzzySubjectSearch(query, constraints, start, end):

Qualified name  KroketApp.query.subject.search.fuzzySubjectSearch

Description  A method implementing the fuzzy Subject Search, the query string will first be stripped of any accents. E.g. é will be replaced with e. All symbols will be removed and the query is split into words. All possible stems are then found using wordnet. Using these stems the KeywordSubjectRelation will be searched for all keywords which contains one of the entered keywords. The KeywordSubjectRelation will then be grouped per subject and the sum of the counts per subject will be calculated. A descending ordering on this sum will be applied and the slice starting at index start and ending at index end will be taken.

Parameters

- query – (String) The searchTerm that should be used to search the database for subjects.
- constraints – (Q) A Q object that will be applied prior on KeywordSubjectRelation prior searching.
- start – (Integer) An integer indicating the starting index of the slice requested.
- end – (Integer) An integer indicating the end index of the slice requested.

Returns  (ValueQuerySet) A ValueQuerySet of the KeywordSubjectRelation with the following fields: subject, subject_code, subject_name, subject_ects, subject_niveau, subject_last_year; rating (which is the sum of all counts in the matching KeywordSubjectRelation,
the higher the number, the better the match is). It will also be descending ordered on rating.

5.14.5 VQSToDict

Declaration  Declared in KroketApp/query/subject/search.py on line 204:
def VQSToDict(vqs):

Qualified name  KroketApp.query.subject.search.VQSToDict

Description  An internal method used only in this script. Converts a ValueQuerySet from KeywordSubjectRelation to a list of dictionaries in standard format.

This ValueQuerySet is related to a list of dictionaries (although not the same, and cannot be converted using simpleJSON). Using this function the subject from the relation is put into the standard format. For performance reasons we do not use the info.standardFormat method, as we can fetch all information in one query, instead of n queries (where n is the amount of subjects found).

Parameters

vqs – (ValueQuerySet) A value query set of the KeywordSubjectRelation with the following fields present: subject_code, subject_name, subject_ects, subject_niveau, subject_lastYear and subject.

Returns  (List) A JSON list of subjects in standardFormat \rightarrow info.standardFormat.
5.15. KROKETAPP.QUERY.USER.AUTHENTICATE

5.15  KroketApp.query.user.authenticate

A query script that handles AJAX requests to /studieplanner/query/user/(register|login|logout). This script will implement multiple interfaces used to register and login/logout. Note that once NT-credentials are used, which is currently not supported, the register function should be deprecated.

In general you are considered logged in, in a view, if you have a key ‘username’ in the request.POST and this corresponds to an account by the field ‘username’ in account. If a username-key is present but no corresponding account, the username-key should be deleted on any request which checks if the user is logged in.

Hierarchical member index  This file contains the following members:

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>167</td>
</tr>
<tr>
<td>TRUE</td>
<td>167</td>
</tr>
<tr>
<td>USERNAME_FORBIDDEN_SYMBOLS</td>
<td>169</td>
</tr>
<tr>
<td>info</td>
<td>169</td>
</tr>
<tr>
<td>isCorrectPassword</td>
<td>170</td>
</tr>
<tr>
<td>isCorrectUsername</td>
<td>170</td>
</tr>
<tr>
<td>is_ascii</td>
<td>170</td>
</tr>
<tr>
<td>login</td>
<td>168</td>
</tr>
<tr>
<td>logout</td>
<td>169</td>
</tr>
<tr>
<td>register</td>
<td>168</td>
</tr>
</tbody>
</table>

Detailed documentation for all declarations in this file follows.

5.15.1  FALSE

Declaration  Declared in KroketApp/query/user/authenticate.py on line 24:
FALSE = False

Qualified name  KroketApp.query.user.authenticate.FALSE

Description  A constant representing a false result value.

5.15.2  TRUE

Declaration  Declared in KroketApp/query/user/authenticate.py on line 29:
TRUE = True

Qualified name  KroketApp.query.user.authenticate.TRUE

Description  A constant representing a true result value.
CHAPTER 5. PYTHON (QUERIES)

5.15.3 register

Declaration  Declared in KroketApp/query/user/authenticate.py on line 50:
def register(request):

Qualified name  KroketApp.query.user.authenticate.register

Description  An AJAX request handler for the ‘register’ function. A JSON dictionary will be returned with one key ‘success’. This key indicates whether a registration was successful or not. Can be TRUE or FALSE.

A registration is successful if and only if all of the following conditions hold:

- the username is not already taken;
- the username is valid; \( \rightarrow \) isCorrectUsername
- the password is valid. \( \rightarrow \) isCorrectPassword

\( \rightarrow \) KroketApp.savemodels.Account.register

Parameters

- username – (String) The requested username of the account.
- password – (String) The requested password of the account.

Returns  (JSON dictionary) A dictionary with a key “success” which has as value TRUE or FALSE, indicating success or failure, respectively.

5.15.4 login

Declaration  Declared in KroketApp/query/user/authenticate.py on line 83:
def login(request):

Qualified name  KroketApp.query.user.authenticate.login

Description  An AJAX request handler for the ‘login’ function. A JSON-dictionary will be returned with one key ‘success’. The value of the key indicates whether the login attempt was a successful or a failure. The value can be either TRUE or FALSE.

If the username is not be valid by isCorrectUsername or the password is not be valid by isCorrectPassword, the result will be success=FALSE.

A login is successful if the database contains an account with as username the parameter username and the password is after ‘salting’ hashed to the same hash as in account. \( \rightarrow \) KroketApp.security.check_credentials

If a user has successfully registered with username u and password p using register the user should be able to login with username u and password p.
5.15. KROKETAPP.QUERY.USER.AUTHENTICATE

Parameters

username – (String) The username with which the user wants to login.
password – (String) The corresponding password with which the user wants to
login.

Returns (JSON-dictionary) A dictionary with a key ‘success’ which has as value TRUE or
FALSE, indicating success or failure, respectively.

5.15.5 logout

Declaration  Declared in KroketApp/query/user/authenticate.py on line 101:
def logout(request):

Qualified name  KroketApp.query.user.authenticate.logout

Description  An AJAX request handler for the ‘logout’ function. Nothing will be returned.
The user will be logout if the user was logged in, otherwise nothing will happen.

5.15.6 info

Declaration  Declared in KroketApp/query/user/authenticate.py on line 114:
def info(request):

Qualified name  KroketApp.query.user.authenticate.info

Description  An AJAX request handler for the ‘info’ function. A dictionary will be returned.
In this dictionary a key ‘isLoggedIn’ is present, which indicates if the user is logged in, can
be TRUE or FALSE. If the user is logged in a key ‘username’ is also present.

Returns (JSON-dict) A dictionary with the login-info.

5.15.7 USERNAME_FORBIDDEN_SYMBOLS

Declaration  Declared in KroketApp/query/user/authenticate.py on line 135:
USERNAME_FORBIDDEN_SYMBOLS = re.compile(r"(\.)|[^a-zA-Z0-9\-\._]")

Qualified name  KroketApp.query.user.authenticate.USERNAME_FORBIDDEN_SYMBOLS

Description  A regular expression which matches any part of a string which is not allowed
in a username. → isCorrectUsername.
CHAPTER 5. PYTHON (QUERIES)

5.15.8 isCorrectUsername

Declaration  Declared in KroketApp/query/user/authenticate.py on line 150:
def isCorrectUsername(username):

Qualified name  KroketApp.query.user.authenticate.isCorrectUsername

Description  Checks whether username is allowed.

Parameters

username – (String) The username, which should be checked.

Returns  (Boolean) True if

- the username only contains letters (A-Z or a-z), numbers (0-9), dashes (-), underscores (_), and periods (.);
- the username contains no more than one period in a row;
- the length of the username is inside the range [5, 20].

Otherwise False is returned.

5.15.9 is_ascii

Declaration  Declared in KroketApp/query/user/authenticate.py on line 158:
def is_ascii(s):

Qualified name  KroketApp.query.user.authenticate.is_ascii

Description  Checks whether a certain string contains only ASCII characters.

Parameters

s – (String) The string to be checked.

Returns  (Boolean) A boolean indicating if all characters are ASCII characters.

5.15.10 isCorrectPassword

Declaration  Declared in KroketApp/query/user/authenticate.py on line 167:
def isCorrectPassword(password):

Qualified name  KroketApp.query.user.authenticate.isCorrectPassword
Description  Checks whether the password is allowed.

Parameters

password – (String) The password to be checked.

Returns  (Boolean) true if the length of the password is at least 8 and all characters are ASCII; false otherwise.
CHAPTER 5. PYTHON (QUERIES)

5.16 KroketApp.query.user.change

A query script that handles AJAX requests to /studieplanner/query/user/change. This script will change several fields of the user accounts.

Hierarchical member index  This file contains the following members:

<table>
<thead>
<tr>
<th>Declaration</th>
<th>Qualified name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>KroketApp.query.user.change.FALSE</td>
<td>A constant representing a false result value.</td>
</tr>
<tr>
<td>TRUE</td>
<td>KroketApp.query.user.change.TRUE</td>
<td>A constant representing a true result value.</td>
</tr>
<tr>
<td>changePassword</td>
<td></td>
<td></td>
</tr>
<tr>
<td>changeProperties</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Detailed documentation for all declarations in this file follows.

5.16.1 FALSE

Declaration  Declared in KroketApp/query/user/change.py on line 16:
FALSE = False

Qualified name  KroketApp.query.user.change.FALSE

Description  A constant representing a false result value.

5.16.2 TRUE

Declaration  Declared in KroketApp/query/user/change.py on line 21:
TRUE = True

Qualified name  KroketApp.query.user.change.TRUE

Description  A constant representing a true result value.

5.16.3 changeProperties

Declaration  Declared in KroketApp/query/user/change.py on line 40:
def changeProperties(request):

Qualified name  KroketApp.query.user.change.changeProperties
5.16. KROKETAPP.QUERY.USER.CHANGE

Description  An AJAX request handler for the 'change properties of account'-function. Arguments will be fetched from request.POST.

This function will change the properties of the account, which is logged in (in the current session). A dictionary will be returned with a key credentials which can be either TRUE or FALSE, indicating if the currentPassword was valid. For each other (allowed) property which was requested a corresponding key is associated with a boolean in the returned dictionary. This boolean indicates if the parameter was valid. If all booleans are TRUE the change was written to the database, otherwise everything was discarded. The user must be logged in.

Parameters

  currentPassword – (String) The current password. This needs to be present to be able to change any property.
  password – (Optional, String) The new password, as a string. This has to be a valid password as defined in user.authentication.isValidPassword.

Returns  (JSON dictionary) A dictionary with a key success indicating if a change was made to the database.

5.16.4 changePassword

Declaration  Declared in KroketApp/query/user/change.py on line 63:
def changePassword(acc, password):

Qualified name  KroketApp.query.user.change.changePassword

Description  A helper method to change the password of a given username. An exception will be raised if the password is not correct. → query.user.authenticate.isCorrectPassword

Parameters

  acc – (Account) The account of which the password should be changed.
  password – (String) The new password, as a string.
Chapter 6

Python (jobs) documentation

In this chapter, all of the Python code for the server side jobs will be documented. Python declarations are ordered on the file they are in.

6.1 KroketApp.schedule.daily_jobs

This is a simple script that runs all daily jobs. This file contains no declarations.
6.2. KroketApp.schedule.daily_jobs.subject_updater.addData

This script adds all sorts of information regarding subjects, majors and scheduling to the database. It also adds several relations between them. Preferably this is done via an OWIS-link, but workarounds like OW-Info are provided since the OWIS-link was not available. We save this information on our server because this makes the whole application faster and more reliable.

Hierarchical member index  This file contains the following members:

- addCourse
- addCoursePlanning
- addSoftwareScienceMajor
- biztalk
- clearDB
- closeFiles
- getAdditionalInfo
- main
- openFiles
- owisCourses
- parseDoelgroep
- parseProgramma
- rankDoelgroep

Detailed documentation for all declarations in this file follows.

6.2.1  biztalk

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 22:

```python
biztalk = TueBizTalk(url = "http://dlwtbiz.campus.test.tue.nl/ESB/ESB_ESB_DLWO_ESB_ReceivePort.svc?wsdl",
```

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.addData.biztalk

Description  This global variable keeps a link with biztalk, a service that communicates with the OWIS database. Accessing the OWIS database can only be done by sending requests to the biztalk-server. More on that in biztalk.py.

6.2.2  openFiles

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 33:

```python
def openFiles():
```

175
CHAPTER 6. PYTHON (JOBS)

Qualified name KroketApp.schedule.daily_jobs.subject_updater.addData.openFiles

Description The first information is given via an excel-file, we saved all that information in three different csv files which we use. An important thing provided by these files are the subject codes of the Bachelor College subjects. There is no other way to get these yet. The same holds for the scheduling of these subjects. This function opens the csv-files, skips the header and converts planning.csv into a list of lines. This is done because several iterations on this file are needed while the rest needs to be read only once.

6.2.3 closeFiles

Declaration Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 49:

```python
def closeFiles():
```

Qualified name KroketApp.schedule.daily_jobs.subject_updater.addData.closeFiles

Description This function closes the two remaining files, which is done after the updating. Closing is done via the global variables they are stored in.

6.2.4 clearDB

Declaration Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 57:

```python
def clearDB():
```

Qualified name KroketApp.schedule.daily_jobs.subject_updater.addData.clearDB

Description This function deletes all information that is currently stored in the database concerning OWIS-information. This is faster and way easier than checking if the data is still up to date.

6.2.5 parseDoelgroep

Declaration Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 73:

```python
def parseDoelgroep():
```

Qualified name KroketApp.schedule.daily_jobs.subject_updater.addData.parseDoelgroep

Description This function uses the doelgroep.csv file and adds the different target_groups. This file only contains the Bachelor College majors and coherent packages. Via a regular expression everything between “(“ and “)“ and a possible space before that is deleted. This is because all the majors have something like “Bachelor College“ in between brackets behind the major-name which is redundant information for our application.
6.2. KROKETAPP.SCHEDULE.DAILY_JOBS.SUBJECT_UPDATER.ADDDATA

6.2.6 addCourse

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 98:
def addCourse(vakcode, level):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.addData.addCourse

Description  This function adds a subject to the database. All subject information is obtained via parser.py, except for level. When a subject is added, keywords need to be added via catagorize.rankEnglish(s). If something goes wrong with parser.py, only the subject is added and hopefully more information is provided via the Biztalk link.

Parameters

vakcode – (String) The subject code of the subject that needs to be added.
level – (String) The difficulty of the subject, preferably one of “Basic”, “Intermediate”, “Advanced” or “”.

6.2.7 addCoursePlanning

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 119:
def addCoursePlanning(wprogramma, lplanning, subj, tg):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.addData.addCoursePlanning

Description  This function adds a planning instance of a subject if the given list of words matches the given line. It could happen that a certain subject is scheduled in two timeslots; in this case additional timeslots are added separated by semicolons.

Parameters

wprogramma – (List of Strings) The list contains strings for all the entities of a programma-line.
lplanning – (String) A line from the planning.csv file that could contain additional info for wprogramma if certain keys are the same.
subj – (Subject) Object of the subject in wprogramma[3].
tg – (Target_group) Object of the target group in wprogramma[1].

6.2.8 parseProgramma

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 136:
def parseProgramma():
CHAPTER 6. PYTHON (JOBS)

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.addData.parseProgramma

Description  This function parses the programa.csv and planning.csv files. These two files are combined and saved in one table. In short this means that if a course is scheduled all information is added to this line. Information like “which major includes this course” is added there as well. In order to create a good table much more information is needed than the files contain. Scheduling contains a subject for example. If this subject is not in the database yet it is added. This does not hold for SubjectPackets like majors and coherent packages.

6.2.9 getAditionalInfo

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 161:

def getAditionalInfo(subj):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.addData.getAditionalInfo

Description  Given a foreign subject key, try to get additional information about the subject from OWIS (Biztalk) and update this information in the database.

Parameters

  subj – (Subject) The subject to get aditional information for.

6.2.10 owisCourses

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 198:

def owisCourses():

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.addData.owisCourses

Description  Here all courses are taken from the database, for every subject that misses one of the fields with additional information, extra information is asked by calling getAditionalInfo with this subject

6.2.11 addSoftwareScienceMajor

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 215:

def addSoftwareScienceMajor():

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.addData.addSoftwareScienceMajor
6.2. KROKETAPP.SCHEDULE.DAILY_JOBS.SUBJECT_UPDATER.ADDDATA

**Description**  This method adds additional subjects to the database.

Subject codes are not known yet for subjects in the second and third year of the majors. Therefore these subjects are not yet in OWIS. For testing purposes, we add in this method all missing subjects of the Software Science major. They have subject codes “KROK00”, “KROK01” and so on. The information about these subjects is taken from the Software Science study guide, but not all information is present there. One course, 2IO90, is already in the database but not yet part of this major and therefore it is added.

Of course, this method should be removed when all information is in OWIS.

6.2.12 rankDoelgroep

**Declaration**  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 254:

def rankDoelgroep():

**Qualified name**  KroketApp.schedule.daily_jobs.subject_updater.addData.rankDoelgroep

**Description**  Assigns keywords to all non-major SubjectPackets in the database.

6.2.13 main

**Declaration**  Declared in KroketApp/schedule/daily_jobs/subject_updater/addData.py on line 265:

def main():

**Qualified name**  KroketApp.schedule.daily_jobs.subject_updater.addData.main

**Description**  The main function calls all the needed functions to add all new information.
CHAPTER 6. PYTHON (JOBS)

6.3 KroketApp.schedule.daily_jobs.subject_updater.biztalk

This file takes care of the communication with biztalk, the interface of the OWIS-database. This file has several parts, namely a class that adds some help to SUDS such that it can make better trees of the returned SOAP message, some Exception classes and a class that deals with the given requests and answers these.

Hierarchical member index  This file contains the following members:

- **BIZTALK_RETURN_TYPEDEF**: 181
- **PatchWSDLPlugin**: 181
- **TueBizTalk**: 183
  - **__init__**: 183
  - **_getClient**: 183
  - **_call**: 184
- **LANGUAGE_NL**: 184
- **Failure**: 184
- **Course**: 185
- **VerzorgendeEenheid**: 185
- **VakVoorkennis**: 185
- **VakVervolgVak**: 185
  - **geefVakGegevens**: 186
- **TueBizTalkException**: 181
- **TueBizTalkInternalError**: 181
- **TueBizTalkMissingAttribute**: 182
- **XSD_PATH**: 180
- **getResultAttr**: 182
- **getResultList**: 182

Detailed documentation for all declarations in this file follows.

6.3.1 XSD_PATH

**Declaration**  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 22:

```python
XSD_PATH = os.path.join(os.path.dirname(__file__), 'xsd')
```

**Qualified name**  KroketApp.schedule.daily_jobs.subject_updater.biztalk.XSD_PATH

**Description**  This is the path to the xsd folder which contains an xsd file with the structures of the SOAP-answers that Biztalk returns.
6.3. KROKETAPP:SCHEDULE.DAILY_JOBS.SUBJECT_UPDATER.BIZTALK

6.3.2 PatchWSDLPlugin

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on
line 29:
class PatchWSDLPlugin(DocumentPlugin):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.PatchWSDLPlugin

Description  A suds DocumentPlugin to patch a WSDL to add additional return types. Changing this shouldn’t be necessary, most likely a .xsd file is missing in the xsd folder then.

6.3.3 BIZTALK_RETURN_TYPEDEF

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on
line 73:

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.BIZTALK_RETURN_TYPEDEF

Description  This list should contain a definition of all possible return types.

6.3.4 TueBizTalkException

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on
line 78:
class TueBizTalkException(Exception):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalkException

Description  Handle biztalk exceptions.

6.3.5 TueBizTalkInternalError

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on
line 84:
class TueBizTalkInternalError(TueBizTalkException):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalkInternalError

Description  Handle internal biztalk errors.
CHAPTER 6. PYTHON (JOBS)

6.3.6 TueBizTalkMissingAttribute

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 94:

class TueBizTalkMissingAttribute(TueBizTalkException):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalkMissingAttribute

Description  Handle missing biztalk attributes.

6.3.7 getResultAttr

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 107:

def getResultAttr(obj, *names):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.getResultAttr

Description  Given an object, return the value in the name attribute of that object.

Parameters

  obj – (Object) The object in which the expected attributes are located.
  *names – (Variable amount of Strings) The names of the attributes that are in the Object of which the values are returned.

Returns  (Object) Returns object.name for a name in *names that is an attribute of obj.

6.3.8 getResultList

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 120:

def getResultList(obj, *names):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.getResultList

Description  Given an object, return the values in name attribute of that object.

Parameters

  obj – (Object) The object in which the expected attributes are located.
  *names – (Variable amount of strings) The names of the attributes that are in Object of which the values are returned.
6.3. KROKETAPP.SCHEDULE.DAILY_JOBS.SUBJECT_UPDATER.BIZTALK

**Returns**  *(Object)* Returns a list of `object.name` for a name in `*names` that is an attribute of `obj`.

6.3.9 TueBizTalk

**Declaration**  Declared in `KroketApp/schedule/daily_jobs/subject_updater/biztalk.py` on line 131:

```python
class TueBizTalk(object):
```

**Qualified name**  `KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk`

**Description**  This class creates SOAP-questions and processes the answers into a workable structure. The first functions are for all biztalk-calls, the last one is specific for the ‘geefVakGegevens’ request.

6.3.10 TueBizTalk.__init__

**Declaration**  Declared in `KroketApp/schedule/daily_jobs/subject_updater/biztalk.py` on line 139:

```python
def __init__(self, url, cache_dir, proxy=None):
```

**Qualified name**  `KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.__init__`

**Description**  Set the values for the communication with the biztalk-server.

**Parameters**

- `url` – *(string)* The URL of the Biztalk server.
- `cache_dir` – *(string)* The directory where requests and responses can be stored.

6.3.11 TueBizTalk._getClient

**Declaration**  Declared in `KroketApp/schedule/daily_jobs/subject_updater/biztalk.py` on line 149:

```python
def _getClient(self):
```

**Qualified name**  `KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk._getClient`

**Description**  This function actually adds the extra information about the answer, as explained in the previous class.
CHAPTER 6. PYTHON (JOBS)

Returns (client) The client where the requests should be sent to.

6.3.12 TueBizTalk._call

Declaration Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 166:
def _call(self, service, **kwargs):

Qualified name KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk._call

Description This function actually creates and sends the request.

Parameters

  service – (string) The name of the needed service.
  **kwargs – (Dictionary of arguments) Tuples of argument names and their value needed for the biztalk-request.

6.3.13 TueBizTalk.LANGUAGE_NL

Declaration Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 198:
LANGUAGE_NL = 'NL'

Qualified name KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.LANGUAGE_NL

Description These are all possible languages in which requests can be sent.

6.3.14 TueBizTalk.Failure

Declaration Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 205:
Failure = namedtuple('Failure', ['code', 'message'])

Qualified name KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.Failure

Description A Failure-answer is a namedtuple consisting of a failure code and its message.
6.3. KROKETAPP.SCHEDULE.DAILY_JOBS.SUBJECT_UPDATER.BIZTALK

6.3.15 TueBizTalk.Course

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 210:
Course = namedtuple('Course', [

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.
Course

Description  A course is returned as a namedtuple with these fieldnames.

6.3.16 TueBizTalk.VerzorgendeEenheid

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 235:
VerzorgendeEenheid = namedtuple('VerzorgendeEenheid', ['Eindverantwoordelijk', 'OgEenheidVrzOms', 'OgEenheidVrzType', 'OgEenheidVrzUrl'])

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.
VerzorgendeEenheid

Description  A VerzorgendeEenheid is returned as a namedtuple with these fieldnames.

6.3.17 TueBizTalk.VakVoorkennis

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 240:
VakVoorkennis = namedtuple('VakVoorkennis', ['VoorkennisInfo', 'VoorkennisVakOmschr', 'VoorkennisVakcode'])

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.
VakVoorkennis

Description  A VakVoorkennis is returned as a namedtuple with these fieldnames.

6.3.18 TueBizTalk.VakVervolgVak

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 245:
VakVervolgVak = namedtuple('VakVervolg', ['VervolgVakInfo', 'VervolgVakVakOmschr', 'VervolgVakVakcode'])

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.
VakVervolgVak

185
CHAPTER 6. PYTHON (JOBS)

Description  A VakVervolgVak is returned as a namedtuple with these fieldnames.

6.3.19  TueBizTalk.geefVakGegevens

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/biztalk.py on line 253:

```python
def geefVakGegevens(self, courseCode, studiejaar_Id=datetime.datetime.now().year, language=LANGUAGE_EN):
```

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.biztalk.TueBizTalk.geefVakGegevens

Description  This function returns the course information from OWIS for a certain course. First the method, language and missing attributes are set. After that the call is made and the result is checked. If there was no error the result is returned as a Course-type which is created there.

Parameters

- `courseCode` - (string) The subject code of the course where more information about is needed.

Returns  (Course) The information of the requested course. If ‘BerichtStatus‘ equals -1, something went wrong and the error is contained in the “Statusomschrijving” field.
6.4 KroketApp.schedule.daily_jobs.subject_updater.parser

This file contains one function which loads a webpage from OW-Info that contains a lot of information for a certain subject. With some help of regular expressions the information of this page is parsed into variables which in the end are added to a subject that is added to the database.

Hierarchical member index  This file contains the following members:

    leesvak ......................................................... 187

Detailed documentation for all declarations in this file follows.

6.4.1 leesvak

Declaration  Declared in KroketApp/schedule/daily_jobs/subject_updater/parser.py on line 16:
def leesvak(vakcode):

Qualified name  KroketApp.schedule.daily_jobs.subject_updater.parser.leesvak

Description  Calling this function actually loads the webpage and parses it. The code is made for temporarily use and should not be used for longer periods of time since its functionality is strongly depending on specific page details in OW-Info.
6.5 KroketApp.schedule.daily_jobs.recommender_job

A class that gets schedules and creates tuples in the RelatedSubject table. This class retrieves all schedules and starts by deleting all irrelevant data; the major courses. When that is done, tuples are created with all combinations of two subjects that are in the schedule. These tuples are put in the global list called tempDB. Because it is likely that there will be double tuples, a check is performed before tuples are added to the RelatedSubject table. This check counts the number of times the same tuple is in the list and deletes these. Not only the subjects are saved, but also the timeslots they are in and the major of the person who choose these subjects. For efficient querying the table is created symmetric.

Hierarchical member index  This file contains the following members:

- createTuples ....................................................... 188
- deleteDoubles .................................................... 189
- deleteOldTable ................................................... 189
- getNrOfDoubles ................................................... 189
- main ................................................................. 189
- tempDB ............................................................. 188

Detailed documentation for all declarations in this file follows.

6.5.1 tempDB

Declaration  Declared in KroketApp/schedule/daily_jobs/recommender_job.py on line 15:
            tempDB= []

Qualified name  KroketApp.schedule.daily_jobs.recommender_job.tempDB

Description  The global tempDB is a list to collect all tuples before they are added to the RelatedSubject table. This is because now a lot of operations are done in the faster main memory of the server.

6.5.2 createTuples

Declaration  Declared in KroketApp/schedule/daily_jobs/recommender_job.py on line 21:
            def createTuples():

Qualified name  KroketApp.schedule.daily_jobs.recommender_job.createTuples

Description  This is a function that that receives all schedules from the database, strips the major subjects from that and creates a list of tuples in the global tempDB list. A tuple consists of two courses, their timeblock and the major that the owner of the schedule takes.
6.5. KROKETAPP.SCHEDULE.DAILY_JOBS.RECOMMENDER_JOB

6.5.3 deleteOldTable

Declaration  Declared in KroketApp/schedule/daily_jobs/recommender_job.py on line 64:
```python
def deleteOldTable():
```

Qualified name  KroketApp.schedule.daily_jobs.recommender_job.deleteOldTable

Description  This function deletes all current objects in the RelatedSubject table.

6.5.4 getNrOfDoubles

Declaration  Declared in KroketApp/schedule/daily_jobs/recommender_job.py on line 76:
```python
def getNrOfDoubles(subject, subj, major):
```

Qualified name  KroketApp.schedule.daily_jobs.recommender_job.getNrOfDoubles

Description  This function searches for tuples in tempDB that are the same as the given tuple. It counts the amount of identical tuples and deletes all these identical tuples such that none are left.

Parameters

- subject – (String) The first subject of the tuple.
- subj – (String) The second subject of the tuple.
- major – (Integer) The identifier (foreign key) of the major the person takes that made the schedule.

Returns  (Integer) The amount of times the same tuple occured in the list tempDB.

6.5.5 deleteDoubles

Declaration  Declared in KroketApp/schedule/daily_jobs/recommender_job.py on line 96:
```python
def deleteDoubles():
```

Qualified name  KroketApp.schedule.daily_jobs.recommender_job.deleteDoubles

Description  This function uses the global variable tempDB to add tuples to the database such that there are no doubles and a rating of the relation is added. The relation rating equals the amount of times the same tuple was in the schedules.

6.5.6 main

Declaration  Declared in KroketApp/schedule/daily_jobs/recommender_job.py on line 119:
```python
def main():
```
CHAPTER 6. PYTHON (JOBS)

Qualified name  KroketApp.schedule.daily_jobs.recommender_job.main

Description  When the daily tasks are run, this function is called. From here all the needed functions are called to make sure the RelatedSubject table is filled with up to date information.
Chapter 7

Unit test documentation

In this chapter, all of the unit test code will be documented. There are both client side (JavaScript) and server side (Python) unit tests.

The unit tests are ordered on the file they are in.

Note that the actual tests, with input and output specifications, are described much more detailed in the UTP (Unit Test Plan). This document only aims to document the methods used for unit testing, while the UTP also is about the contents of the tests.

7.1 test/base-test.js

This file contains some basic tests to verify that the testing environment is working properly.

Function index  This file contains the following functions:

    basicTest()  ................................................................. 191

Detailed documentation for all declarations in this file follows.

7.1.1 basicTest()

Declaration  Declared in test/base-test.js on line 8:
    function basicTest() {

Description  Some simple tests, to verify that the testing environment is working properly.
CHAPTER 7. UNIT TESTS

7.2 test/studyplanner-test.js

Tests the functions in studyplanner.js.

Function index  This file contains the following functions:

- testAddRemoveYear() ........................................ 192
- testCheckDoubleCourses() .................................. 193
- testColorCourses() ........................................... 193
- testEmptySchedule() .......................................... 193
- testFillInCourses() ........................................... 192
- testRemoveCourse() ........................................... 193
- testSaveMajor() .............................................. 193
- testSetRecommendedSubjects() .............................. 194
- testSetYear() .................................................. 192
- testShowSubjectInformation() ............................... 193
- testUpdateMajorErrorBar() ................................. 192

Detailed documentation for all declarations in this file follows.

7.2.1 testSetYear()

Declaration Declared in test/studyplanner-test.js on line 8:
function testSetYear() {

Description  Tests the setYear function, which changes the year a student started his or her study.

7.2.2 testAddRemoveYear()

Declaration  Declared in test/studyplanner-test.js on line 18:
function testAddRemoveYear() {

Description  Tests the addYear and removeYear functions.

7.2.3 testFillInCourses()

Declaration  Declared in test/studyplanner-test.js on line 39:
function testFillInCourses() {

Description  Tests fillInCourses.

7.2.4 testUpdateMajorErrorBar()

Declaration  Declared in test/studyplanner-test.js on line 69:
function testUpdateMajorErrorBar() {
7.2. TEST/STUDYPLANNER-TEST.JS

Description Tests the updateMajorErrorBar function. (Also calls setMajor.)

7.2.5 testSaveMajor()

Declaration Declared in test/studyplanner-test.js on line 95:
function testSaveMajor() {

Description Tests the saveMajor function.

7.2.6 testColorCourses()

Declaration Declared in test/studyplanner-test.js on line 116:
function testColorCourses() {

Description Tests colorCourses. Also uses fillInCourses.

7.2.7 testRemoveCourse()

Declaration Declared in test/studyplanner-test.js on line 147:
function testRemoveCourse() {

Description Test removeCourse (by calling the event handler on the remove icon).

7.2.8 testShowSubjectInformation()

Declaration Declared in test/studyplanner-test.js on line 162:
function testShowSubjectInformation() {

Description Tests showSubjectInformation. (Does not test showing the description of a course, this is done by another function).

7.2.9 testCheckDoubleCourses()

Declaration Declared in test/studyplanner-test.js on line 185:
function testCheckDoubleCourses() {

Description Small test of checkDoubleCourses. Note that this function mainly uses checkDoubleCoursesRecursive, which is tested properly.

7.2.10 testEmptySchedule()

Declaration Declared in test/studyplanner-test.js on line 212:
function testEmptySchedule() {

Description Tests emptySchedule.
CHAPTER 7. UNIT TESTS

7.2.11  testSetRecommendedSubjects()

Declaration  Declared in test/studyplanner-test.js on line 236:
function testSetRecommendedSubjects() {

Description  Tests setRecommendedSubjects.
7.3 test/load-test.js

This file contains the unit tests for the functions in load.js.

**Function index**  This file contains the following functions:

- `testAskingMajors()` ................................. 195
- `testAskingMajorsMocking()` ....................... 195
- `testBuildTable()` ................................. 195

Detailed documentation for all declarations in this file follows.

7.3.1 testAskingMajorsMocking()

**Declaration**  Declared in test/load-test.js on line 9:

```javascript
function testAskingMajorsMocking() {

```

**Description**  Tests the asking of all majors by faking a server connection (‘mocking’).

7.3.2 testAskingMajors()

**Declaration**  Declared in test/load-test.js on line 46:

```javascript
function testAskingMajors() {

```

**Description**  Tests the asking of all majors to the actual server.

7.3.3 testBuildTable()

**Declaration**  Declared in test/load-test.js on line 101:

```javascript
function testBuildTable() {

```

**Description**  Tests the build table function.
CHAPTER 7. UNIT TESTS

7.4 test/authentication-test.js

This file contains tests for the functions in authentication.js.

Function index  This file contains the following functions:

    testValidPassword() .......................................................... 196
    testValidUsername() ......................................................... 196

Detailed documentation for all declarations in this file follows.

7.4.1 testValidUsername()

Declaration  Declared in test/authentication-test.js on line 20:
function testValidUsername() {

Description  Test the isValidUsername(username) (see page 52) function. Input with expected result:

- SomeName - This is a valid username.
- Some Name - This is not a valid username, it contains a space.
- SomeName123 - This is also a valid username.
- Some-Name - Also a valid username.
- SomeName* - Not a valid username, it contains a *.
- Names - A valid username.
- Name - Not a valid username (too short).
- ThisNamelnReallyLong - A valid username.
- ThisReallyIsALongName - Not a valid username (too long).

7.4.2 testValidPassword()

Declaration  Declared in test/authentication-test.js on line 47:
function testValidPassword() {

Description  Test the isValidPassword(password) (see page 53) function. Input with expected result:

- password - Should be considered valid.
- passwrd - Not valid (too short).
- Thisisaverylongpasswordthatisreallyreallylong - Valid.

196
7.4. TEST AUTHENTICATION-TEST.JS

- h - Not valid (extended ASCII range).
- containsatab\t - Not valid, ASCII control characters are not allowed.
CHAPTER 7. UNIT TESTS

7.5 test/validator-test.js

This file contains tests for the functions in validator.js.

Function index  This file contains the following functions:

- testCheckCoherent() .................................................. 198
- testCheckMajor() ..................................................... 198
- testCheckOverlap() .................................................. 198
- testCheckUse() ....................................................... 198

Detailed documentation for all declarations in this file follows.

7.5.1 testCheckMajor()

Declaration  Declared in test/validator-test.js on line 8:
function testCheckMajor() {

Description  Tests the checkMajor(data) (3.9.3) function.

7.5.2 testCheckUse()

Declaration  Declared in test/validator-test.js on line 21:
function testCheckUse() {

Description  Tests the checkUse(data) (3.9.4) function.

7.5.3 testCheckCoherent()

Declaration  Declared in test/validator-test.js on line 34:
function testCheckCoherent() {

Description  Tests the checkCoherent(data) (3.9.5) function.

7.5.4 testCheckOverlap()

Declaration  Declared in test/validator-test.js on line 47:
function testCheckOverlap() {

Description  Tests the checkOverlap(data) (3.9.2) function.
This file contains the unit tests for the functions in utils.js.

Function index  This file contains the following functions:

```plaintext
isEq(a1, a2) .................................................. 201
testCheckDoubleCoursesRecursive() ................................ 200
testDisableCourses() ........................................... 199
testFilterTimeSlot() ............................................. 200
testGetSchedule() ............................................... 200
testHighlight() .................................................. 200
testMakePackage() ............................................... 201
testMakeSubject() ............................................... 201
testNoConflict() ............................................... 201
testNoCurrConflict() ........................................... 201
testRemoveHighlight() .......................................... 200
testSetModified() ............................................... 199
testSetScheduleLoading() ..................................... 199
testSetSubjectInfo() ........................................... 200
```

Detailed documentation for all declarations in this file follows.

7.6.1 testSetScheduleLoading()

Declaration  Declared in test/utils-test.js on line 22:

```javascript
function testSetScheduleLoading() {
```

Description  Tests the showDisabler and hideDisabler functions.

7.6.2 testSetModified()

Declaration  Declared in test/utils-test.js on line 44:

```javascript
function testSetModified() {
```

Description  Tests the setModified function. This does not test if the onbeforeunload event is set properly. There is no proper way to check this. (You could go to the website and try yourself, of course.)

7.6.3 testDisableCourses()

Declaration  Declared in test/utils-test.js on line 55:

```javascript
function testDisableCourses() {
```

Description  Tests the disabeling of courses
CHAPTER 7. UNIT TESTS

7.6.4 testFilterTimeSlot()

Declaration  Declared in test/utils-test.js on line 78:
function testFilterTimeSlot(){

Description  Checks the filter timeslot function, which should change a ;";" in the word "and"

7.6.5 testGetSchedule()

Declaration  Declared in test/utils-test.js on line 87:
function testGetSchedule(){

Description  Tests the getSchedule function. First runs a test with an empty schedule, then with a schedule with 1 course in year 2 quartile 3 (note this is the 7th quartile if you don't consider years).

7.6.6 testSetSubjectInfo()

Declaration  Declared in test/utils-test.js on line 123:
function testSetSubjectInfo(){

Description  Tests the setSubjectInfo function (which shows some subject info in the GUI)

7.6.7 testHighlight()

Declaration  Declared in test/utils-test.js on line 135:
function testHighlight(){

Description  Tests the highlight function. Note this function also uses buildTable and setYear!

7.6.8 testRemoveHighlight()

Declaration  Declared in test/utils-test.js on line 149:
function testRemoveHighlight(){

Description  Tests the removeHighlight function.

7.6.9 testCheckDoubleCoursesRecursive()

Declaration  Declared in test/utils-test.js on line 163:
function testCheckDoubleCoursesRecursive(){

Description  Tests the checkDoubleCoursesRecursive function.
7.6. TEST/UTILS-TEST.JS

7.6.10 testMakePackage()

Declaration  Declared in test/utils-test.js on line 196:
function testMakePackage(){

Description  Tests the makePackage function.

7.6.11 testMakeSubject()

Declaration  Declared in test/utils-test.js on line 209:
function testMakeSubject(){

Description  Tests the makeSubject function.

7.6.12 testNoConflict()

Declaration  Declared in test/utils-test.js on line 228:
function testNoConflict(){

Description  Tests the noConflict function (help function of checkDoubleCourses)

7.6.13 testNoCurrConflict()

Declaration  Declared in test/utils-test.js on line 239:
function testNoCurrConflict(){

Description  Tests the noCurrConflict function (help function of noConflict)

7.6.14 isEq(a1, a2)

Declaration  Declared in test/utils-test.js on line 254:
function isEq(a1, a2){

Description  Checks if a1 and a2 are equal, where a1 and a2 are 2-dimensional arrays.
CHAPTER 7. UNIT TESTS

7.7 test/storage-test.js

This file contains the unit tests for the functions in storage.js.

Function index  This file contains the following functions:

  testSaveName() .................................................. 202
  testShowSchedule() ............................................. 202

Detailed documentation for all declarations in this file follows.

7.7.1 testSaveName()

Declaration  Declared in test/storage-test.js on line 9:
            function testSaveName() {

Description  Test that when clicking Save and the current schedule name is empty, the user is asked for a save name.

7.7.2 testShowSchedule()

Declaration  Declared in test/storage-test.js on line 39:
            function testShowSchedule() {

Description  Test the showSchedule(data) function, and alongside that function also many other functions are tested.
7.8 test/searchSubjects-test.js

This file contains the unit tests for the functions in searchSubjects.js.

**Function index**  This file contains the following functions:

- testGetPostdataSearch() ................................................ 203
- testKeywordHighlight() .................................................. 203
- testRemoveKeywordHighlight() ........................................ 203
- testShowSearchResults() .................................................. 203

Detailed documentation for all declarations in this file follows.

### 7.8.1 testGetPostdataSearch()

**Declaration**  Declared in test/searchSubjects-test.js on line 8:

```javascript
function testGetPostdataSearch()
```

**Description**  Tests if the postdata from search() is correct. May not work in IE, because we simulate click events and...well....IE seems to have another function to do this.

### 7.8.2 testShowSearchResults()

**Declaration**  Declared in test/searchSubjects-test.js on line 41:

```javascript
function testShowSearchResults()
```

**Description**  Tests the showSearchResults function.

### 7.8.3 testKeywordHighlight()

**Declaration**  Declared in test/searchSubjects-test.js on line 53:

```javascript
function testKeywordHighlight()
```

**Description**  Tests the keywordHighlight function.

### 7.8.4 testRemoveKeywordHighlight()

**Declaration**  Declared in test/searchSubjects-test.js on line 66:

```javascript
function testRemoveKeywordHighlight()
```

**Description**  Tests the removeKeywordHighlight function.
CHAPTER 7. UNIT TESTS

7.9 test/searchPackages-test.js

This file contains test for the functionality of searchPackage.js

Function index This file contains the following functions:

- testCoherentSearchButtonWaiting() .................................. 204
- testShowCoherentPackages() ........................................ 204

Detailed documentation for all declarations in this file follows.

7.9.1 testCoherentSearchButtonWaiting()

Declaration Declared in test/searchPackages-test.js on line 8:
function testCoherentSearchButtonWaiting(){

Description Tests the function coherentSearchButtonWaiting

7.9.2 testShowCoherentPackages()

Declaration Declared in test/searchPackages-test.js on line 19:
function testShowCoherentPackages(){

Description Tests the showCoherentPackages function.

7.10 KroketApp.tests.addData_tester

This class tests the module addData.py on a unit testing level. To make testing possible some additional classes are imported. First the unittest-class from Django which basically adds the same functionality as the unittest-class of Python. Further we need to prepare some data so we need to be able to query, therefore the models are imported. addData and parser are two classes we want to test and categorize is also needed to fill the database with correct data (after a course is added its keywords need to be find).

Hierarchical member index This file contains the following members:

- testAddData ......................................................... 205
- test_ClearDB .................................................... 205
- test_ParseDoelgroep ............................................. 205
- test_AddCourse ................................................. 205
- test_AddCoursePlanning ........................................ 206
- test_ParseProgramma .......................................... 206
- test_GetAditionalInfo .......................................... 206
- test_DwisCourses ............................................. 206
- test_AddSoftwareScienceMajor ............................... 207

Detailed documentation for all declarations in this file follows.
7.10. KROKETAPP.TESTS.ADDDATA_TESTER

7.10.1 testAddData

Declaration  Declared in KroketApp/tests/addData_tester.py on line 13:
class testAddData(unittest.TestCase):

Qualified name  KroketApp.tests.addData_tester.testAddData

Description  This class actually tests the addData.py

7.10.2 testAddData.test_ClearDB

Declaration  Declared in KroketApp/tests/addData_tester.py on line 26:
def test_ClearDB(self):

Qualified name  KroketApp.tests.addData_tester.testAddData.test_ClearDB

Description  To make sure the delete function works, first some items are added to the database. This is to make sure an already empty table doesn’t always conclude the deleting went correctly. Therefore it is also important that after adding the data a check is done that there is indeed at least one item in the table. After deleting it is only checked if there are no items left in the tables, if so everything is deleted.

7.10.3 testAddData.test_ParseDoelgroep

Declaration  Declared in KroketApp/tests/addData_tester.py on line 73:
def test_ParseDoelgroep(self):

Qualified name  KroketApp.tests.addData_tester.testAddData.test_ParseDoelgroep

Description  To test if parsing the doelgroep.csv did work we take 3 known items in that file and check if these are added correctly.

7.10.4 testAddData.test_AddCourse

Declaration  Declared in KroketApp/tests/addData_tester.py on line 92:
def test_AddCourse(self):

Qualified name  KroketApp.tests.addData_tester.testAddData.test_AddCourse

Description  Testing if a course is added correctly is done by trying two random courses. A third, non-existing course, is also tried but this should just add the course with no information.
CHAPTER 7. UNIT TESTS

7.10.5  testAddData.test_AddCoursePlanning

**Declaration**  Declared in *KroketApp/tests/addData_tester.py* on line 119:
```python
def test_AddCoursePlanning(self):
```

**Qualified name**  KroketApp.tests.addData_tester.testAddData.test_AddCoursePlanning

**Description**  This function does two tests, the first one should be correct and the second wrong. The preparation done before the tests is similar to what is done during the processing of the files so the needed parameters are taken directly from these files. The variables lprogramma and lplanning are copied lines from the files. In the failing case these two do not match.

7.10.6  testAddData.test_ParseProgramma

**Declaration**  Declared in *KroketApp/tests/addData_tester.py* on line 163:
```python
def test_ParseProgramma(self):
```

**Qualified name**  KroketApp.tests.addData_tester.testAddData.test_ParseProgramma

**Description**  Here it is tested if the importing of *Programma.csv* is done correctly. One known object is checked to be true.

7.10.7  testAddData.test_GetAdditionalInfo

**Declaration**  Declared in *KroketApp/tests/addData_tester.py* on line 176:
```python
def test_GetAdditionalInfo(self):
```

**Qualified name**  KroketApp.tests.addData_tester.testAddData.test_GetAdditionalInfo

**Description**  This tests the link with OWIS, *geefVakGegevens*. for one subject more info is asked from the server and checked to be the same as expected.

7.10.8  testAddData.test_OwisCourses

**Declaration**  Declared in *KroketApp/tests/addData_tester.py* on line 192:
```python
def test_OwisCourses(self):
```

**Qualified name**  KroketApp.tests.addData_tester.testAddData.test_OwisCourses

**Description**  A subject is added with several blanks. If everything is ok, the subject is been processed and more info is added.
7.10.9 testAddData.test_addSoftwareScienceMajor

Declaration  Declared in KroketApp/tests/addData_tester.py on line 210:
```python
def test_addSoftwareScienceMajor(self):
```

Qualified name  KroketApp.tests.addData_tester.testAddData.test_addSoftwareScienceMajor

Description  The function addSoftwareScienceMajor adds a set of major-courses which are hardcoded. To test this, the function is run and after that it is checked if one of those subjects is correctly in the database. One course is already in the database and is only added to the major, this is checked as well.
7.11 KroketApp.tests.recommender_job_tester

This class tests the module recommender_job.py on a unit-testing level. The first imported class gives certain unit-test features, pretty much the same as Python’s unit testing environment. The second imported class is the class that is tested. Models is imported so we can do our own querying here to prepare certain datasets. The same holds for the classes addData and parser, they are used for adding needed data.

Hierarchical member index This file contains the following members:

- testRecommender_job ........................................... 208
- setUp .............................................................. 208
- test_CreateTuples .............................................. 208
- test_DeleteOldTable ........................................... 209
- test_GetNrOfDoubles ........................................... 209
- test_DeleteDoubles ............................................. 209

Detailed documentation for all declarations in this file follows.

7.11.1 testRecommender_job

Declaration Declared in KroketApp/tests/recommender_job_tester.py on line 15:
class testRecommender_job(unittest.TestCase):

Qualified name KroketApp.tests.recommender_job_tester.testRecommender_job

Description This is the class uses by Django to test recommender_job.py. The function setUp(self) is been called before every test is run, further there is a function, named test_[FunctionNameFrom_recommender_job.py] for every function in recommender_job.py.

7.11.2 testRecommender_job.setUp

Declaration Declared in KroketApp/tests/recommender_job_tester.py on line 21:
def setUp(self):

Qualified name KroketApp.tests.recommender_job_tester.testRecommender_job.setUp

Description Before the tests can be run, all old data is removed and the global variables are defined. These global variables are added such that there is data for all functions that need data. self.l is a list that can be assigned to addData.tempDB. It should be symmetric and can contain more tuples of the same.

7.11.3 testRecommender_job.test_CreateTuples

Declaration Declared in KroketApp/tests/recommender_job_tester.py on line 72:
def test_CreateTuples(self):
Qualified name  KroketApp.tests.recommender_job_tester.testRecommender_job.test_CreateTuples

Description  This will test the function createTuples(), but can only be done if it is known which tuples have to be added. This is done by making sure the given schema’s are known. Therefore several courses, a major and a schema is added to the database. Major 1365 contains the course oLABo, so this shouldn’t appear in the database, the others do.

When everything is added, the function that we want to test is run and afterwards some checks are performed if the correct items are added to tempDB and if several logical tuples are not in there.

7.11.4 testRecommender_job.test_DeleteOldTable

Declaration  Declared in KroketApp/tests/recommender_job_tester.py on line 116:
def test_DeleteOldTable(self):

Qualified name  KroketApp.tests.recommender_job_tester.testRecommender_job.test_DeleteOldTable

Description  To make sure the delete function works, first some items are added to the database. This is to make sure an already empty table doesn’t always conclude the deleting went correctly. Therefore it is also important that after adding the data a check is done that there is indeed at least one item in the table. After deleting it is only checked if there are no items left in the table, if so all are deleted.

7.11.5 testRecommender_job.test_GetNrOfDoubles

Declaration  Declared in KroketApp/tests/recommender_job_tester.py on line 130:
def test_GetNrOfDoubles(self):

Qualified name  KroketApp.tests.recommender_job_tester.testRecommender_job.test_GetNrOfDoubles

Description  Here is checked if getNrOfDoubles does find the correct amount of doubles by using the predefined global variables. The last line checks if all items are in deed deleted after counting them.

7.11.6 testRecommender_job.test_DeleteDoubles

Declaration  Declared in KroketApp/tests/recommender_job_tester.py on line 146:
def test_DeleteDoubles(self):

Qualified name  KroketApp.tests.recommender_job_tester.testRecommender_job.test_DeleteDoubles
CHAPTER 7. UNIT TESTS

**Description**  After `deleteDoubles()` is executed there should be several results in the RelatedSubjects table. First we add the known predefined set of courses and then we check if the function `getRelatedSubjects(subj1)` indeed returns the correct courses. The last check is a tuple that should not be in there.
7.12. KroketApp.tests.testCategorize

This file tests the KroketApp/categorize.py file. Invalid requests and valid requests are tested. We use the subjectTestSet to load model instances in the database. → KroketApp.categorize

Hierarchical member index This file contains the following members:

- `html_tags` ................................................................. 212
- `keyword_test_set` ....................................................... 211
- `subjects` ................................................................. 211
- `testPacket` ............................................................... 214
  - `setUp` ................................................................. 214
  - `tearDown` ............................................................. 214
  - `test_rankPacket` .................................................... 215
- `testSubject` ............................................................. 213
  - `setUp` ................................................................. 213
  - `tearDown` ............................................................. 213
  - `test_rankEnglish` .................................................... 213
  - `test_updateSubDB` ................................................... 214
- `test_rankStringNLTK` .................................................. 212
  - `setUp` ................................................................. 212
  - `tearDown` ............................................................. 212

Detailed documentation for all declarations in this file follows.

7.12.1 subjects

Declaration Declared in KroketApp/tests/testCategorize.py on line 15:

```
subjects = ['0HV30', '2IT60', '4MA00', '5EIA0', '8QA01', '2AS00']
```

Qualified name KroketApp.tests.testCategorize.subjects

Description A list of subject codes that are in the database. These are used as test set for our testcases.

7.12.2 keyword_test_set

Declaration Declared in KroketApp/tests/testCategorize.py on line 21:

```
keyword_test_set = {
```

Qualified name KroketApp.tests.testCategorize.keyword_test_set

Description A dictionary mapping a SubjectPacket by target group identifier to a set of expected keywords. This dictionary also maps a Subject by subject code to a set of expected keywords.
CHAPTER 7. UNIT TESTS

7.12.3 html_tags

Declaration  Declared in KroketApp/tests/testCategorize.py on line 37:
html_tags = ['<p>', '</p>', '<ul>', '</ul>', '<li>', '</li>', '<div>', '</div>',

Qualified name  KroketApp.tests.testCategorize.html_tags

Description  Some HTML tags that should filtered out of the remarks by categorize and should never be a keyword.

7.12.4 testrankStringNLTK

Declaration  Declared in KroketApp/tests/testCategorize.py on line 44:
class testrankStringNLTK(unittest.TestCase):

Qualified name  KroketApp.tests.testCategorize.testrankStringNLTK

Description  A testcase that tests the function: KroketApp.categorize.rankStringNLTK. → KroketApp.categorize.rankStringNLTK

7.12.5 testrankStringNLTK.setUp

Declaration  Declared in KroketApp/tests/testCategorize.py on line 50:
def setUp(self):

Qualified name  KroketApp.tests.testCategorize.testrankStringNLTK.setUp

Description  Performs some initialization tasks. Is empty in this testcase. Is called before a call to every test_function.

7.12.6 testrankStringNLTK.tearDown

Declaration  Declared in KroketApp/tests/testCategorize.py on line 56:
def tearDown(self):

Qualified name  KroketApp.tests.testCategorize.testrankStringNLTK.tearDown

Description  Performs some destruction tasks. Is empty in this testcase. Is called after every call to a test_

7.12.7 testrankStringNLTK.test_rankStringNLTK

Declaration  Declared in KroketApp/tests/testCategorize.py on line 63:
def test_rankStringNLTK(self):

212
7.12. KROKETAPP.TESTS.TESTCATEGORY

**Qualified name**  KroketApp.tests.testCategorize.testrankStringNLTK.test_rankStringNLTK

**Description**  Makes some calls to KroketApp.categorize.rankStringNLTK with some testing sentences. → KroketApp.categorize.rankStringNLTK

### 7.12.8 testSubject

**Declaration**  Declared in KroketApp/tests/testCategorize.py on line 89:

class testSubject(unittest.TestCase):

**Qualified name**  KroketApp.tests.testCategorize.testSubject

**Description**  This class actually tests the function KroketApp.categorize.rankEnglish (which is completely identical to KroketApp.categorize.rankSubject. Also tests KroketApp.categorize.updateSubDB. → KroketApp.categorize.rankSubject→ KroketApp.categorize.rankEnglish → KroketApp.categorize.updateSubDB

### 7.12.9 testSubject.setUp

**Declaration**  Declared in KroketApp/tests/testCategorize.py on line 95:

def setUp(self):

**Qualified name**  KroketApp.tests.testCategorize.testSubjectsetUp

**Description**  Loads data in the database. Is called before a call to every test function.

### 7.12.10 testSubject.tearDown

**Declaration**  Declared in KroketApp/tests/testCategorize.py on line 115:

def tearDown(self):

**Qualified name**  KroketApp.tests.testCategorize.testSubject.tearDown

**Description**  Performs some destruction tasks. Clears the database. Is called after every call to a test_

### 7.12.11 testSubject.test_rankEnglish

**Declaration**  Declared in KroketApp/tests/testCategorize.py on line 130:

def test_rankEnglish(self):

**Qualified name**  KroketApp.tests.testCategorize.testSubject.test_rankEnglish
CHAPTER 7. UNIT TESTS

Description Makes a call to KroketApp.categorize.rankEnglish for every subject in the database. Then checks using keyword_test_set if the subject is associated with the expected keywords.

7.12.12 testSubject.test_updateSubDB

Declaration Declared in KroketApp/tests/testCategorize.py on line 151:

def test_updateSubDB(self):

Qualified name KroketApp.tests.testCategorize.testSubject.test_updateSubDB

Description Makes a call to KroketApp.categorize.updateSubDB for every subject in the database. We use as arguments a dictionary with as keys the expected keys for each subject as defined in keyword_test_set. We then check if the function updateSubDB has correctly associated the keywords with the subject.

7.12.13 testPacket

Declaration Declared in KroketApp/tests/testCategorize.py on line 169:
class testPacket(unittest.TestCase):

Qualified name KroketApp.tests.testCategorize.testPacket

Description This class actually tests the function KroketApp.categorize.rankPacket.

7.12.14 testPacket.setUp

Declaration Declared in KroketApp/tests/testCategorize.py on line 174:
def setUp(self):

Qualified name KroketApp.tests.testCategorize.testPacket.setUp

Description Loads data in the database. Also categorizes all subjects in the database with categorize.rankEnglish/ Is called before a call to every test_function.

7.12.15 testPacket.tearDown

Declaration Declared in KroketApp/tests/testCategorize.py on line 195:
def tearDown(self):

Qualified name KroketApp.tests.testCategorize.testPacket.tearDown
Description  Performs some destruction tasks. Clears the database. Is called after every call to a test_.

7.12.16  testPacket.test_rankPacket

Declaration  Declared in KroketApp/tests/testCategorize.py on line 210:
    def test_rankPacket(self):

Qualified name  KroketApp.tests.testCategorize.testPacket.test_rankPacket

Description  Makes a call to KroketApp.categorize.rankPacket for every (non Major) SubjectPacket in the database. Then checks using keyword_test_set if the SubjectPacket is associated with the expected keywords.
CHAPTER 7. UNIT TESTS

7.13 KroketApp.tests.testSecurity

A Python script that partially tests the functionality of security.py. Other parts of security.py are covered in other tests, as they use the methods used for login and logout.

Hierarchical member index  This file contains the following members:

    test_credentials ............................................ 216

Detailed documentation for all declarations in this file follows.

7.13.1 test_credentials

Declaration  Declared in KroketApp/tests/testSecurity.py on line 36:
def test_credentials(self):

Qualified name  KroketApp.tests.testSecurity.test_credentials

Description  Test whether the account exists on the database. The account exists only if both the username and the password is correct. A few accounts have been made by using some combinations of usernames and passwords. However, every combination of username and password will be tested.
7.14.  KROKETAPP.TESTS.QUERY.MAJOR.TESTMAJOR

7.14  KroketApp.tests.query.major.testMajor

A Python script that tests the methods of /query/major/list.py and /query/major/subjects.py.

Hierarchical member index  This file contains the following members:

    test_list ................................................. 217
    test_package ........................................... 217

Detailed documentation for all declarations in this file follows.

7.14.1  test_list

Declaration  Declared in KroketApp/tests/query/major/testMajor.py on line 95:
            def test_list(self):

Qualified name  KroketApp.tests.query.major.testMajor.test_list

Description  Test for the list procedure. Multiple packets with different names have been added to the database. A few packets have been added to a subset of these packets. It tests whether it loads all major packets from the database.

7.14.2  test_package

Declaration  Declared in KroketApp/tests/query/major/testMajor.py on line 107:
            def test_package(self):

Qualified name  KroketApp.tests.query.major.testMajor.test_package

Description  Test for the subjects procedure. Multiple packets with different names have been added to the database. A few packets have been added to a subset of these packets. It tests whether the packet returns the right subjects which have been added to this packet, and no other subjects.
CHAPTER 7. UNIT TESTS

7.15 KroketApp.tests.query.package.testPackage

A Python script that tests the methods of /query/package/search.py and /query/package/subjects.py.

Hierarchical member index  This file contains the following members:

- test_package ......................................................... 218
- test_search .......................................................... 218

Detailed documentation for all declarations in this file follows.

7.15.1 test_search

Declaration  Declared in KroketApp/tests/query/package/testPackage.py on line 107:
def test_search(self):

Qualified name  KroketApp.tests.query.package.testPackage.test_search

Description  Test for the search procedure. Multiple packets with different names and remarks have been added to the database. Several search phrases and their expected results have been predefined and will be used to test. These combinations test on words whether these words exist in remarks or name of the package. Substring of a word are also tested. Common words like “you” should not trigger any package.

7.15.2 test_package

Declaration  Declared in KroketApp/tests/query/package/testPackage.py on line 120:
def test_package(self):

Qualified name  KroketApp.tests.query.package.testPackage.test_package

Description  Test for subjects in a package. Multiple packets with different names and remarks have been added to the database. A package is tested whether it contains the subject enlisted to it and to the length of the packet, since there are subjects which are not enlisted to the package.
7.16. KROKETAPP.TESTS.QUERY.RECOMMENDATION.TESTRECOMMENDATION

7.16  KroketApp.tests.query.recommendation.testRecommendation

A Python script that tests the methods of /query/recommendation/subject.py.

Hierarchical member index  This file contains the following members:

    test_recommendations .................................................. 219

Detailed documentation for all declarations in this file follows.

7.16.1  test_recommendations

Declaration  Declared in KroketApp/tests/query/recommendation/testRecommendation.py on line 165:

def test_recommendations(self):

Qualified name  KroketApp.tests.query.recommendation.testRecommendation.test_recommendations

Description  Test for the recommendation of subjects. Several schedules have been created. These schedules may contain major, elective or USE subjects, a combation of them, or all. The procedure should not make recommendations based on major subjects of a schedule, but rather on the elective and USE subjects. This functionality will be tested.
CHAPTER 7. UNIT TESTS

7.17 KroketApp.tests.query.schedule.testSchedule

A Python script that tests methods of /query/schedule/save.py, /query/schedule/load.py, /query/schedule/delete.py and /query/schedule/validate.py.

Hierarchical member index This file contains the following members:

- test_all_schedules ..................................................... 221
- test_delete_schedule .................................................. 221
- test_load_non_existing_schedule ................................. 220
- test_load_non_related_schedule ................................. 220
- test_load_schedule .................................................. 220
- test_rename_schedule ................................. 221
- test_overwrite_schedule ........................................ 221

Detailed documentation for all declarations in this file follows.

7.17.1 test_load_schedule

Declaration  Declared in KroketApp/tests/query/schedule/testSchedule.py on line 76:
def test_load_schedule(self):

Qualified name  KroketApp.tests.query.schedule.testSchedule.test_load_schedule

Description  Test for the save and load procedure. An account is created and used to login. It creates a schedule, saves it, logs out, logs in and tries to open the same schedule. The major, begin year and list of subjects (also the placement in the right time blocks) should be the same.

7.17.2 test_load_non_related_schedule

Declaration  Declared in KroketApp/tests/query/schedule/testSchedule.py on line 113:
def test_load_non_related_schedule(self):

Qualified name  KroketApp.tests.query.schedule.testSchedule.test_load_non_related_schedule

Description  Test for the save and load procedure. Two accounts are created, from which one account creates a schedule and saves it, the other account tries to load that schedule. It should be unable to load that account.

7.17.3 test_load_non_existing_schedule

Declaration  Declared in KroketApp/tests/query/schedule/testSchedule.py on line 149:
def test_load_non_existing_schedule(self):
7.17. KROKETAPP.TESTS.QUERY.SCHEDULE.TESTSCHEDULE

**Qualified name** KroketApp.tests.query.schedule.testSchedule.test_load_non_existing_schedule

**Description** Test for the save and load procedure. One account is created, it saves a schedule and then logs out. Afterwards it logs in again, but tries to load a schedule with a different name (non-existing one), which should fail.

7.17.4 test_overwrite_schedule

**Declaration** Declared in KroketApp/tests/query/schedule/testSchedule.py on line 181:

def test_overwrite_schedule(self):

**Qualified name** KroketApp.tests.query.schedule.testSchedule.test_overwrite_schedule

**Description** Test for overwriting. One account is created and save a schedule. It relogs and saves a new schedule with the same name as the previous one. The old schedule should be overwritten.

7.17.5 test_delete_schedule

**Declaration** Declared in KroketApp/tests/query/schedule/testSchedule.py on line 230:

def test_delete_schedule(self):

**Qualified name** KroketApp.tests.query.schedule.testSchedule.test_delete_schedule

**Description** Test for deletion of the schedule. One account is created, and it saves two schedules as two different names. One of these schedules is deleted, so the account should only have schedule left.

7.17.6 test_rename_schedule

**Declaration** Declared in KroketApp/tests/query/schedule/testSchedule.py on line 276:

def test_rename_schedule(self):

**Qualified name** KroketApp.tests.query.schedule.testSchedule.test_rename_schedule

**Description** Test for renaming a schedule. An account is created. Two schedules are created, from which one will be renamed.

7.17.7 test_all_schedules

**Declaration** Declared in KroketApp/tests/query/schedule/testSchedule.py on line 330:

def test_all_schedules(self):
CHAPTER 7. UNIT TESTS

Qualified name  KroketApp.tests.query.schedule.testSchedule.test_all_schedules

Description  Test to retrieve all schedules. One account is created, and two schedules are saved. This procedure should return both of these schedules.
7.18 KroketApp.tests.query.schedule.testValidate

This file tests the query/schedule/validate.py file. Invalid requests and valid requests are tested. → KroketApp.query.schedule.validate

Hierarchical member index  This file contains the following members:

<table>
<thead>
<tr>
<th>CONTENT_TYPE</th>
<th>223</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoherentTestCase</td>
<td>228</td>
</tr>
<tr>
<td>setUp</td>
<td>229</td>
</tr>
<tr>
<td>tearDown</td>
<td>229</td>
</tr>
<tr>
<td>test_pass</td>
<td>229</td>
</tr>
<tr>
<td>test_fail</td>
<td>229</td>
</tr>
<tr>
<td>OverlapTestCase</td>
<td>227</td>
</tr>
<tr>
<td>setUp</td>
<td>227</td>
</tr>
<tr>
<td>tearDown</td>
<td>228</td>
</tr>
<tr>
<td>test_pass</td>
<td>228</td>
</tr>
<tr>
<td>test_fail</td>
<td>228</td>
</tr>
<tr>
<td>USETestCase</td>
<td>230</td>
</tr>
<tr>
<td>setUp</td>
<td>230</td>
</tr>
<tr>
<td>tearDown</td>
<td>230</td>
</tr>
<tr>
<td>test_pass</td>
<td>230</td>
</tr>
<tr>
<td>test_fail</td>
<td>230</td>
</tr>
<tr>
<td>ValidateTestCase</td>
<td>224</td>
</tr>
<tr>
<td>setUp</td>
<td>224</td>
</tr>
<tr>
<td>tearDown</td>
<td>225</td>
</tr>
<tr>
<td>assertValidJSON_and_Load</td>
<td>225</td>
</tr>
<tr>
<td>test_invalidRequest</td>
<td>225</td>
</tr>
<tr>
<td>test_passAll</td>
<td>225</td>
</tr>
<tr>
<td>test_failOverlap</td>
<td>226</td>
</tr>
<tr>
<td>test_failCoherent</td>
<td>226</td>
</tr>
<tr>
<td>test_failUSE</td>
<td>226</td>
</tr>
<tr>
<td>test_failMajor</td>
<td>227</td>
</tr>
<tr>
<td>clearData</td>
<td>224</td>
</tr>
<tr>
<td>loadData</td>
<td>224</td>
</tr>
</tbody>
</table>

Detailed documentation for all declarations in this file follows.

7.18.1 CONTENT_TYPE

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 17:
CONTENT_TYPE = 'application/json'

Qualified name  KroketApp.tests.query.schedule.testValidate.CONTENT_TYPE
CHAPTER 7. UNIT TESTS

Description  A content type as string identifying the expected content type for a JSON response.

7.18.2  loadData

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 23:
def loadData():

Qualified name  KroketApp.tests.query.schedule.testValidate.loadData

Description  Generates subjects, packages, majors and some overlapping packets. Saves the created instances of the models in the database.

7.18.3  clearData

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 101:
def clearData():

Qualified name  KroketApp.tests.query.schedule.testValidate.clearData

Description  Clears all data generated by loadData → loadData

7.18.4  ValidateTestCase

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 110:
class ValidateTestCase(unittest.TestCase):

Qualified name  KroketApp.tests.query.schedule.testValidate.ValidateTestCase

Description  A testcase that tests the function: KroketApp.query.schedule.validate. → KroketApp.query.schedule.validate

7.18.5  ValidateTestCase.setUp

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 117:
def setUp(self):

Qualified name  KroketApp.tests.query.schedule.testValidate.ValidateTestCase.setUp

Description  Performs some initialization tasks. Creates a client that allows the tests to make HTTPRequests. Calls loadData to initialize the database. Is called before a call to every test_function. → loadData

224
7.18. KROKETAPP.TESTS.QUERY.SCHEDULE.TESTVALIDATE

7.18.6 ValidateTestCase.tearDown

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 125:

```python
def tearDown(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.ValidateTestCase.tearDown

Description  Performs some destruction tasks. Clears the database by calling clearData.
Is called after every call to a test. → clearData

7.18.7 ValidateTestCase.assertValidJSON_and_Load

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 132:

```python
def assertValidJSON_and_Load(self, response, msg_identifier):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.ValidateTestCase.assertValidJSON_and_Load

Description  Checks with assertions that response is in correct JSON-format and returns a deserialization to a Python object of the response, if possible.

7.18.8 ValidateTestCase.test_invalidRequest

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 145:

```python
def test_invalidRequest(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.ValidateTestCase.test_invalidRequest

Description  Sends some invalid requests for each major to KroketApp.query.schedule.validate.validate and checks if the function correctly returns a 403-forbidden status code.

7.18.9 ValidateTestCase.test_passAll

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 176:

```python
def test_passAll(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.ValidateTestCase.test_passAll
CHAPTER 7. UNIT TESTS

Description   For a certain major m, we brute-force on combinations of coherent packages of length 2 to 3 and use packages of length 1 to 2. With the restriction that the packages have no subject in common. The major parameter will be the target group identifier of m and the schedule parameter will be the set of all mandatory subjects belonging to the major, and the set of all subjects belonging to the use/coherent packages. Furthermore, all headers and methods are as required.

7.18.10 ValidateTestCase.test_failOverlap

Declaration   Declared in KroketApp/tests/query/schedule/testValidate.py on line 232:
def test_failOverlap(self):

Qualified name KroketApp.tests.query.schedule.testValidate.ValidateTestCase.test_failOverlap

Description   For a certain major m, we brute-force on combinations of coherent packages of length 2 to 3 and use packages of length 1 to 2. With the restriction that some package have subjects in common. The major parameter will be the target group identifier of m and the schedule parameter will be the set of all mandatory subjects belonging to the major, and the set of all subjects belonging to the use/coherent packages. Furthermore, all headers and methods are as required.

7.18.11 ValidateTestCase.test_failCoherent

Declaration   Declared in KroketApp/tests/query/schedule/testValidate.py on line 298:
def test_failCoherent(self):

Qualified name KroketApp.tests.query.schedule.testValidate.ValidateTestCase.test_failCoherent

Description   For a certain major m, we brute-force on combinations of coherent packages of length 0 to 1 and use packages of length 1 to 2. With the restriction that the packages have no subject in common. The major parameter will be the target group identifier of m and the schedule parameter will be the set of all mandatory subjects belonging to the major, and the set of all subjects belonging to the use/coherent packages. Furthermore, all headers and methods are as required.

7.18.12 ValidateTestCase.test_failUSE

Declaration   Declared in KroketApp/tests/query/schedule/testValidate.py on line 354:
def test_failUSE(self):

Qualified name KroketApp.tests.query.schedule.testValidate.ValidateTestCase.test_failUSE

226
Description  For a certain major \( m \), we brute-force on combinations of coherent packages of length 2 to 3 and no use packages. With the restriction that the packages have no subject in common. The major parameter will be the target group identifier of \( m \) and the schedule parameter will be the set of all mandatory subjects belonging to the major, and the set of all subjects belonging to the use/coherent packages. Furthermore, all headers and methods are as required.

7.18.13 ValidateTestCase.test_failMajor

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 406:
```
def test_failMajor(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.ValidateTestCase.test_failMajor

Description  For a certain major \( m \), we brute-force on combinations of coherent packages of length 2 to 3 and use packages of length 1 to 2. With the restriction that the packages have no subject in common. We then brute-force on all subsets of the major courses, with the extra requirement that all but 3 major courses are chosen in the schedule, furthermore we include in the schedule parameter the set of all subjects belonging to the use/coherent packages. Furthermore, all headers and methods are as required.

7.18.14 OverlapTestCase

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 467:
```
class OverlapTestCase(unittest.TestCase):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.OverlapTestCase

Description  A testcase that tests the function:
```
→ KroketApp.query.schedule.checkOverlap
```

7.18.15 OverlapTestCase.setUp

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 474:
```
def setUp(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.OverlapTestCase.setUp

Description  Performs some initialization tasks. Loads data in the database. Is called before a call to every test function.
CHAPTER 7. UNIT TESTS

7.18.16 OverlapTestCase.tearDown

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 481:
```python
def tearDown(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.OverlapTestCase.tearDown

Description  Performs some destruction tasks. Clears the database. Is called after every call to a test..

7.18.17 OverlapTestCase.test_pass

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 488:
```python
def test_pass(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.OverlapTestCase.test_pass

Description  Performs a test by brute forcing on all non-overlapping packages. All responses should have a status of ‘PASS’.

7.18.18 OverlapTestCase.test_fail

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 505:
```python
def test_fail(self):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.OverlapTestCase.test_fail

Description  Performs a test by brute forcing on all combinations of packages where at least two packages have on subject in common. All Responses should have a status of ‘FAIL’.

7.18.19 CoherentTestCase

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 542:
```python
class CoherentTestCase(unittest.TestCase):
```

Qualified name  KroketApp.tests.query.schedule.testValidate.CoherentTestCase

Description  A testcase that tests the function: KroketApp.query.schedule.checkCoherent. → KroketApp.query.schedule.checkCoherent
7.18. KROKETAPP_TESTS_QUERY_SCHEDULE_TESTVALIDATE

7.18.20  CoherentTestCase setUp

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 549:
  def setUp(self):

Qualified name  KroketApp.tests.query.schedule.testValidate.CoherentTestCase.setUp

Description  Performs some initialization tasks. Loads data in the database. Is called before a call to every test function.

7.18.21  CoherentTestCase tearDown

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 556:
  def tearDown(self):

Qualified name  KroketApp.tests.query.schedule.testValidate.CoherentTestCase.tearDown

Description  Performs some destruction tasks. Clears the database. Is called after every call to a test.

7.18.22  CoherentTestCase test pass

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 563:
  def test_pass(self):

 Qualified name  KroketApp.tests.query.schedule.testValidate.CoherentTestCase.test_pass

Description  Performs a test by brute forcing on all coherent packages, with as extra requirement that we have chosen at least 2 packages. All responses should have a status of 'PASS'.

7.18.23  CoherentTestCase test fail

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 582:
  def test_fail(self):

 Qualified name  KroketApp.tests.query.schedule.testValidate.CoherentTestCase.test_fail

Description  Performs a test by brute forcing on all use packages, with as extra requirement that we have chosen at most 1 coherent package. All responses should have a status of 'FAIL'.
CHAPTER 7. UNIT TESTS

7.18.24 USETestCase

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 607:
class USETestCase(unittest.TestCase):

Qualified name  KroketApp.tests.query.schedule.testValidate.USETestCase

Description  A testcase that tests the function: KroketApp.query.schedule.checkUSE. →
KroketApp.query.schedule.checkUSE

7.18.25 USETestCase.setUp

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 614:
def setUp(self):

Qualified name  KroketApp.tests.query.schedule.testValidate.USETestCase.setUp

Description  Performs some initialization tasks. Loads data in the database. Is called before
a call to every test. function.

7.18.26 USETestCase.tearDown

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 622:
def tearDown(self):

Qualified name  KroketApp.tests.query.schedule.testValidate.USETestCase.tearDown

Description  Performs some destruction tasks. Clears the database. Is called after every
call to a test_

7.18.27 USETestCase.test_pass

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 629:
def test_pass(self):

Qualified name  KroketApp.tests.query.schedule.testValidate.USETestCase.test_pass

Description  Performs a test by brute forcing on all use packages, with as extra requirement
that we have chosen at least 1 USE package. All responses should have a status of ‘PASS’.

7.18.28 USETestCase.test_fail

Declaration  Declared in KroketApp/tests/query/schedule/testValidate.py on line 648:
def test_fail(self):

230
7.18. KROKETAPP.TESTS.QUERY.SCHEDULE.TESTVALIDATE

Qualified name  KroketApp.tests.query.schedule.testValidate.USETestCase.test_fail

Description  Performs a test by brute forcing on all use packages, with as extra requirement that we have chosen no USE packages. All responses should have a status of ‘FAIL’.
CHAPTER 7. UNIT TESTS

7.19  KroketApp.tests.query.subject.subjectTestSet

This file contains some functions that create test data for several other test files.

Hierarchical member index  This file contains the following members:

- def getPriorKnowledge
- getFollowUp
- getPlannings
- getSubjectPackage
- getTestSet
- sub
- tg

Detailed documentation for all declarations in this file follows.

7.19.1  getTestSet

Declaration  Declared in KroketApp/tests/query/subject/subjectTestSet.py on line 13:

```python
def getTestSet():
```

Qualified name  KroketApp.tests.query.subject.subjectTestSet.getTestSet

Description  Returns a dictionary of subjects, for testing purposes.

Returns  (Dictionary) The test data.

7.19.2  sub

Declaration  Declared in KroketApp/tests/query/subject/subjectTestSet.py on line 187:

```python
def sub(code):
```

Qualified name  KroketApp.tests.query.subject.subjectTestSet.sub

Description  Helper method that finds a subject with a certain subject code. The behaviour when the code could not be found in the database is unspecified.

Parameters

- String – (code) The subject code to look for.

Returns  (Subject) The subject found.
7.19. KROKETAPP.TESTS.QUERY.SUBJECT.SUBJECTTESTSET

7.19.3  
tg

Declaration  Declared in KroketApp/tests/query/subject/subjectTestSet.py on line 197:
def tg(target_group):

Qualified name  KroketApp.tests.query.subject.subjectTestSet.tg

Description  Helper method that finds a subject packet with a certain target_group identifier. The behaviour when the target_group could not be found in the database is unspecified.

Parameters

    Integer – (target_group) The target_group to look for.

Returns  (SubjectPacket) The subject packet found.

7.19.4  
getSubjectPackage

Declaration  Declared in KroketApp/tests/query/subject/subjectTestSet.py on line 204:
def getSubjectPackage():

Qualified name  KroketApp.tests.query.subject.subjectTestSet.getSubjectPackage

Description  Returns a list of subject packages, for testing purposes.

Returns  (List of SubjectPacket) The test data.

7.19.5  
def getPriorKnowledge

Declaration  Declared in KroketApp/tests/query/subject/subjectTestSet.py on line 219:
def getPriorKnowledge():

Qualified name  KroketApp.tests.query.subject.subjectTestSet.def getPriorKnowledge

Description  Returns a list of prior-knowledge relations, for testing purposes.

Returns  (List of SubjectPriorKnowledgeRelation) The test data.

7.19.6  
getFollowUp

Declaration  Declared in KroketApp/tests/query/subject/subjectTestSet.py on line 230:
def getFollowUp():
CHAPTER 7. UNIT TESTS

Qualified name  KroketApp.tests.query.subject.subjectTestSet.getFollowUp

Description  Returns a list of follow-up relations, for testing purposes.

Returns  (List of SubjectFollowUpRelation) The test data.

7.19.7 getPlannings

Declaration  Declared in KroketApp/tests/query/subject/subjectTestSet.py on line 240:
def getPlannings():

Qualified name  KroketApp.tests.query.subject.subjectTestSet.getPlannings

Description  Returns a list of course plannings, for testing purposes.

Returns  (List of CoursePlanning) The test data.
7.20. KROKETAPP.TESTS.QUERY.SUBJECT.TESTINFO

7.20  KroketApp.tests.query.subject.testInfo

This file tests the query/subject/info.py file. Invalid requests and valid requests are tested.
We use the subjectTestSet to load model instances in the database. → KroketApp.query.subject.info

Hierarchical member index  This file contains the following members:

```
CONTENT_TYPE .................................................. 235
ConvertTestCase .................................................. 240
    setUp .................................................. 240
    tearDown .......................................... 240
assertValidJSON_and_Load ...................................... 241
test_invalidRequest .......................................... 241
test_validRequest ........................................... 241
InfoTestCase ................................................... 238
    FIELDS .............................................. 238
    planning .......................................... 238
    follow_up ......................................... 238
    prior_knowledge .................................... 239
    setUp .............................................. 239
    tearDown .......................................... 239
assertValidJSON_and_Load ...................................... 239
test_invalidRequest .......................................... 239
test_validRequest ........................................... 239
equal_lists .................................................. 236
equal_subject ............................................... 237
expec ...................................................... 237
planning .................................................... 237
prior_knowledge ............................................. 236
unsorted_equal ............................................. 236
```

Detailed documentation for all declarations in this file follows.

7.20.1  CONTENT_TYPE

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 19:
CONTENT_TYPE = 'application/json'

Qualified name  KroketApp.tests.query.subject.testInfo.CONTENT_TYPE

Description  A content type as string identifying the expected content type for a JSON response.
CHAPTER 7. UNIT TESTS

7.20.2 unsorted_equal

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 27:
def unsorted_equal(list1, list2):

Qualified name  KroketApp.tests.query.subject.testInfo.unsorted_equal

Description  Checks if the two lists are equal ignoring the ordering. This is a shallow comparison.

Parameters

List – (list1) The list which should be compared with list2.
List – (list2) The list which should be compared with list1.

Returns  (Boolean) A boolean indicating if the lists are equal to each other.

7.20.3 equal_lists

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 38:
def equal_lists(test_list, expect_list):

Qualified name  KroketApp.tests.query.subject.testInfo.equal_lists

Description  Tests if arbitrary nested list of subjects in standard form is equal to an arbitrary nested list of expect subjects. These expected subjects are different, as is explained in expec. This is a deep comparison. → expec

Parameters

List – (test_list) The list which should be tested against expect_list.

Returns  (Boolean) A boolean indicating if the tested list is expected as defined by the expect_list.

7.20.4 prior_knowledge

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 54:
prior_knowledge = {‘0HV30’:[’’],

Qualified name  KroketApp.tests.query.subject.testInfo.prior_knowledge
7.20. KROKETAPP.TESTS.QUERY.SUBJECT.TESTINFO

Description  A dictionary associating to each subject what prior knowledge fields are expected in the result of query/subject/info.

7.20.5 planning

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 65:
planning = {‘0HV30’:[{‘year’:’1’, ‘quartile’:’2’, ‘timeSlot’:’A’, ‘target_group’: 6},

Qualified name  KroketApp.tests.query.subject.testInfo.planning

Description  A dictionary associating to each subject what course plannings should be in the result of query/subject/info.

7.20.6 expec

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 93:
def expec(code):

Qualified name  KroketApp.tests.query.subject.testInfo.expec

Description  Constructs a dictionary determining what standard formats are allowable. The returned subject determines what standard format in the following way: ‘code’, ‘name’, ‘ects’, ‘difficulty’, ‘lastYear’ should be equal. ‘planning’ should be equal without respecting the ordering. The field ‘priorKnowledge’ of a standard format should be in the list given by the ‘priorKnowledge’ of the expected subject.

Parameters

String – (code) The subject code of the subject from which you want to have the expected standard format

Returns  (Dictionary) A dictionary determining the expected standard format.

7.20.7 equal_subject

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 114:
def equal_subject(test_sub, expect_sub):

Qualified name  KroketApp.tests.query.subject.testInfo.equal_subject
CHAPTER 7. UNIT TESTS

Description Checks whether the given subject is the same as the expected subject. This is done in the following way:

- ‘code’, ‘name’, ‘ects’, ‘difficulty’, ‘lastYear’ should be equal.
- ‘planning’ should be equal without respecting the ordering.
- The field ‘priorKnowledge’ of a standard format should be in the list given by the ‘priorKnowledge’ of the expected subject.

7.20.8 InfoTestCase

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 135:
class InfoTestCase(unittest.TestCase):

Qualified name KroketApp.tests.query.subject.testInfo.InfoTestCase

Description A testcase that tests the function: KroketApp.query.subject.info.info, and any related functions. → KroketApp.query.subject.info.info

7.20.9 InfoTestCase.FIELDS

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 141:
FIELDS = ['code', 'name', 'ects', 'difficulty', 'lastYear', 'remarks', 'studyGoal',]

Qualified name KroketApp.tests.query.subject.testInfo.InfoTestCase.FIELDS

Description The fields that should be present in the extended format.

7.20.10 InfoTestCase.planning

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 148:
planning = {'0HV30':{'year':'1', 'quartile':'2', 'timeSlot':'A', 'target_group': 6,

Qualified name KroketApp.tests.query.subject.testInfo.InfoTestCase.planning

Description A dictionary mapping a subject code to the expected planning for that subject.

7.20.11 InfoTestCase.follow_up

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 166:
follow_up = {'0HV30':[]},

Qualified name KroketApp.tests.query.subject.testInfo.InfoTestCase.follow_up

Description A dictionary mapping a subject code to the allowed follow_up strings.
7.20. KROKETAPP.TESTS.QUERY.SUBJECT.TESTINFO

7.20.12 InfoTestCase.prior_knowledge

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 176:

```
prior_knowledge = {'0HV30':[],
```

Qualified name  KroketApp.tests.query.subject.testInfo.InfoTestCase.prior_knowledge

Description  A dictionary mapping a subject code to the allowed follow_up strings.

7.20.13 InfoTestCase.setUp

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 188:

```
def setUp(self):
```

Qualified name  KroketApp.tests.query.subject.testInfo.InfoTestCase.setUp

Description  Performs some initialization tasks. Loads data in the database and creates a test client that can be used to make HTTPRequests to the server. Is called before a call to every test_function.

7.20.14 InfoTestCase.tearDown

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 210:

```
def tearDown(self):
```

Qualified name  KroketApp.tests.query.subject.testInfo.InfoTestCase.tearDown

Description  Performs some destruction tasks. Clears the database. Is called after every call to a test_.

7.20.15 InfoTestCase.assertValidJSON_and_Load

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 221:

```
def assertValidJSON_and_Load(self, response, msg_identifier):
```

Qualified name  KroketApp.tests.query.subject.testInfo.InfoTestCase.assertValidJSON_and_Load

Description  Checks with assertions that response is in correct JSON-format and returns a deserialization to a Python object of the response, if possible.

7.20.16 InfoTestCase.test_invalidRequest

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 234:

```
def test_invalidRequest(self):
```
CHAPTER 7. UNIT TESTS

Qualified name KroketApp.tests.query.subject.testInfo.InfoTestCase.test_invalidRequest

Description Sends some invalid requests to KroketApp.query.subject.info.info and checks if the function correctly returns a 403-forbidden status code.

7.20.17 InfoTestCase.test_validRequest

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 257:
def test_validRequest(self):

Qualified name KroketApp.tests.query.subject.testInfo.InfoTestCase.test_validRequest

Description Performs for every object in the database a request to KroketApp.query.subject.info.info and performs some checks on the returned response.

7.20.18 ConvertTestCase

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 314:
class ConvertTestCase(unittest.TestCase):

Qualified name KroketApp.tests.query.subject.testInfo.ConvertTestCase

Description A testcase that tests the function: KroketApp.query.subject.info.converty, and any related functions.

7.20.19 ConvertTestCase.setUp

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 320:
def setUp(self):

Qualified name KroketApp.tests.query.subject.testInfo.ConvertTestCase.setUp

Description Loads data in the database and creates a test client that can be used to make HTTPRequests to the server. Is called before a call to every test function.

7.20.20 ConvertTestCase.tearDown

Declaration Declared in KroketApp/tests/query/subject/testInfo.py on line 355:
def tearDown(self):

Qualified name KroketApp.tests.query.subject.testInfo.ConvertTestCase.tearDown

Description Clears the database. Is called after every call to a test.
ConvertTestCase.assertValidJSON_and_Load

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 366:
def assertValidJSON_and_Load(self, response, msg_identifier):

Qualified name  KroketApp.tests.query.subject.testInfo.ConvertTestCase.assertValidJSON_and_Load

Description  Checks with assertions that response is in correct JSON-format and returns a deserialization to a Python object of the response, if possible.

ConvertTestCase.test_invalidRequest

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 379:
def test_invalidRequest(self):

Description  Sends some invalid requests to KroketApp.query.subject.info.convert and checks if the function correctly returns a 403-forbidden status code.

ConvertTestCase.test_validRequest

Declaration  Declared in KroketApp/tests/query/subject/testInfo.py on line 399:
def test_validRequest(self):

Description  Performs for some test lists a request to KroketApp.query.subject.info.convert and performs some checks on the returned response.
CHAPTER 7. UNIT TESTS

7.21 KroketApp.tests.query.subject.testSearch

This file tests the query/subject/search.py file. Invalid requests and valid requests are tested. We use the subjectTestSet to load model instances in the database. → KroketApp.query.subject.subject

Hierarchical member index  This file contains the following members:

CONTENT_TYPE .................................................. 242
SearchTestCase  .................................................... 245
FIELDS .............................................................. 245
planning ............................................................ 245
follow_up ........................................................... 245
prior_knowledge .................................................... 246
setUp ............................................................... 246
tearDown ............................................................ 246
assertValidJSON_and_Load  ................................. 246
test_invalidRequest ........................................... 247
test_BasicSearchCodeTerm ..................................... 247
test_SearchTermFormat ........................................... 247
test_range .......................................................... 247
test_quartile ......................................................... 248
test_timeSlot ......................................................... 248
test_year ............................................................ 248
test_difficulty ....................................................... 248
test_targetGroup ..................................................... 249
equal_lists .......................................................... 243
equal_subject ........................................................ 244
expec .............................................................. 244
planning ............................................................ 244
prior_knowledge .................................................... 243
unsorted_equal ..................................................... 243

Detailed documentation for all declarations in this file follows.

7.21.1 CONTENT_TYPE

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 19:

CONTENT_TYPE = 'application/json'

Qualified name  KroketApp.tests.query.subject.testSearch.CONTENT_TYPE

Description  A content type as string identifying the expected content type for a JSON response.
7.21. KROKETAPP.TESTS.QUERY.SUBJECT.TESTSEARCH

7.21.2 unsorted_equal

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 27:
def unsorted_equal(list1, list2):

Qualified name  KroketApp.tests.query.subject.testSearch.unsorted_equal

Description  Checks if the two lists are equal ignoring the ordering. This is a shallow comparison.

Parameters

List – (list1) The list which should be compared with list2.
List – (list2) The list which should be compared with list1.

Returns  (Boolean) A boolean indicating if the lists are equal to each other.

7.21.3 equal_lists

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 38:
def equal_lists(test_list, expect_list):

Qualified name  KroketApp.tests.query.subject.testSearch.equal_lists

Description  Tests if arbitrary nested list of subjects in standard form is equal to an arbitrary nested list of expect subjects. These expected subjects are different, as is explained in expect. This is a deep comparison. → expect

Parameters

List – (test_list) The list which should be tested against expect_list.

Returns  (Boolean) A boolean indicating if the tested list is expected as defined by the expect_list.

7.21.4 prior_knowledge

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 54:

prior_knowledge = {'0HV30': [''],

Qualified name  KroketApp.tests.query.subject.testSearch.prior_knowledge
CHAPTER 7. UNIT TESTS

Description  A dictionary associating to each subject what prior knowledge fields are expected in the result of query/subject/info.

7.21.5  planning

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 65:
planning = {'0HV30': [{'year': '1', 'quartile': '2', 'timeSlot': 'A', 'target_group': 6},

Qualified name  KroketApp.tests.query.subject.testSearch.planning

Description  A dictionary associating to each subject what course plannings should be in the result of query/subject/info.

7.21.6  expec

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 93:
def expec(code):

Qualified name  KroketApp.tests.query.subject.testSearch.expec

Description  Constructs a dictionary determining what standard formats are allowable. The returned subject determines what standard format in the following way: ‘code’, ‘name’, ‘ects’, ‘difficulty’, ‘lastYear’ should be equal. ‘planning’ should be equal without respecting the ordering. The field ‘priorKnowledge’ of a standard format should be in the list given by the ‘priorKnowledge’ of the expected subject.

Parameters

String – (code) The subject code of the subject from which you want to have the expected standard format

Returns  (Dictionary) A dictionary determining the expected standard format.

7.21.7  equal_subject

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 114:
def equal_subject(test_sub, expect_sub):

Qualified name  KroketApp.tests.query.subject.testSearch.equal_subject
Description Checks whether the given subject is the same as the expected subject. This is done in the following way:

- ‘code’, ‘name’, ‘ects’, ‘difficulty’, ‘lastYear’ should be equal. ‘planning’ should be equal without respecting the ordering. The field ‘priorKnowledge’ of a standard format should be in the list given by the ‘priorKnowledge’ of the expected subject.

7.21.8 SearchTestCase

Declaration Declared in KroketApp/tests/query/subject/testSearch.py on line 135:

class SearchTestCase(unittest.TestCase):

Qualified name KroketApp.tests.query.subject.testSearch.SearchTestCase

Description A testcase that tests the function KroketApp.query.subject.search.subject, and any related functions. → KroketApp.query.subject.search.subject

7.21.9 SearchTestCase.FIELDS

Declaration Declared in KroketApp/tests/query/subject/testSearch.py on line 140:

FIELDS = ['code', 'name', 'ects', 'difficulty', 'lastYear', 'remarks', 'studyGoal',

Qualified name KroketApp.tests.query.subject.testSearch.SearchTestCase.FIELDS

Description The fields that should be present in the extended format.

7.21.10 SearchTestCase.planning

Declaration Declared in KroketApp/tests/query/subject/testSearch.py on line 147:

planning = {'0HV30': [{'year': '1', 'quartile': '2', 'timeSlot': 'A', 'target_group': 6},

Qualified name KroketApp.tests.query.subject.testSearch.SearchTestCase.planning

Description A dictionary mapping a subject code to the expected planning for that subject.

7.21.11 SearchTestCase.follow_up

Declaration Declared in KroketApp/tests/query/subject/testSearch.py on line 166:

follow_up = {'0HV30': [''],

Qualified name KroketApp.tests.query.subject.testSearch.SearchTestCase.follow_up
CHAPTER 7. UNIT TESTS

Description  A dictionary mapping a subject code to the allowed follow_up strings.

7.21.12 SearchTestCase.prior_knowledge

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 176:

```python
prior_knowledge = {'0HV30': ['']},
```

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.prior_knowledge

Description  A dictionary mapping a subject code to the allowed follow_up strings.

7.21.13 SearchTestCase.setUp

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 189:

```python
def setUp(self):
```

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.setUp

Description  Performs some initialization tasks. Loads data in the database and creates a test client that can be used to make HTTP requests to the server. Is called before a call to every test_function. Also creates a list of search term and the expected response.

7.21.14 SearchTestCase.tearDown

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 219:

```python
def tearDown(self):
```

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.tearDown

Description  This method performs some destruction tasks. Clears the database. Is called after every call to a test_.

7.21.15 SearchTestCase.assertValidJSON_and_Load

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 233:

```python
def assertValidJSON_and_Load(self, response, msg_identifier):
```

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.assertValidJSON_and_Load

Description  Checks with assertions that response is in correct JSON format and returns a deserialization to a Python object of the response, if possible.
7.21. KROKETAPP.TESTS.QUERY.SUBJECT.TESTSEARCH

7.21.16 SearchTestCase.test_invalidRequest

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 246:
def test_invalidRequest(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_invalidRequest

Description  Sends some invalid requests to KroketApp.query.subject.search.subject and checks if the function correctly returns a 403-forbidden status code.

7.21.17 SearchTestCase.test_BasicSearchCodeTerm

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 263:
def test_BasicSearchCodeTerm(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_BasicSearchCodeTerm

Description  Performs for every object in the database a request to KroketApp.query.subject.search.subject. Gives as only parameter 'searchTerm' with as value the subject code.

7.21.18 SearchTestCase.test_SearchTermFormat

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 276:
def test_SearchTermFormat(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_SearchTermFormat

Description  Performs some searches by sending a request to KroketApp.query.subject.search.subject. Sends different values of 'searchTerm' and compares the result with the expected result.

7.21.19 SearchTestCase.test_range

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 296:
def test_range(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_range

Description  Performs some searches by sending a request to KroketApp.query.subject.search.subject. Sends different values of 'range'. Checks if the response is a 403-Forbidden if the value of the parameter are in the wrong format. Checks if amount results is correctly limited by the range.
7.21.20 SearchTestCase.test_quartile

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 325:
def test_quartile(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_quartile

Description  Performs some searches by sending a request to KroketApp.query.subject.
search.subject. Sends different values of ‘quartile’. Compares the result with the
expected result.

7.21.21 SearchTestCase.test_timeSlot

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 343:
def test_timeSlot(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_timeSlot

Description  Performs some searches by sending a request to KroketApp.query.subject.
search.subject. Sends different values of ‘timeSlot’. Compares the result with the
expected result.

7.21.22 SearchTestCase.test_year

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 363:
def test_year(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_year

Description  Performs some searches by sending a request to KroketApp.query.subject.
search.subject. Sends different values of ‘year’. Compares the result with the expected
result.

7.21.23 SearchTestCase.test_difficulty

Declaration  Declared in KroketApp/tests/query/subject/testSearch.py on line 383:
def test_difficulty(self):

Qualified name  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_difficulty

Description  Performs some searches by sending a request to KroketApp.query.subject.
search.subject. Sends different values of ‘difficulty’. Compares the result with the
expected result.
7.21. KROKETAPP.TESTS.QUERY.SUBJECT.TESTSEARCH

7.21.24 SearchTestCase.test_targetGroup

**Declaration**  Declared in KroketApp/tests/query/subject/testSearch.py on line 400:
```python
def test_targetGroup(self):
```

**Qualified name**  KroketApp.tests.query.subject.testSearch.SearchTestCase.test_targetGroup

**Description**  Performs some searches by sending a request to KroketApp.query.subject.search.subject. Sends different values of 'target_group'. Compares the result with the expected result.
7.22 KroketApp.tests.query.user.testAuthenticate

A Python script that tests the methods of /query/user/authenticate.py.

Hierarchical member index  This file contains the following members:

LoginTestCase ............................................ 250
denyAllLogins ........................................... 250
assertValidJSON_and_Load ............................... 255
test_ValidLogin ......................................... 251
test_InvalidLoginTest ................................. 251
assertValidJSON_and_Load ............................... 255
test_InvalidPost ................................. 252
test_BasicLogouts ...................................... 253
assertValidJSON_and_Load ............................... 255
test_ValidCredentialsRegisters ....................... 253
test_WrongUsernamePasswordRegisters ................. 254
test_InvalidRequests .................................. 254
assertValidJSON_and_Load ............................... 255
test_LoggedUserInfo .................................... 255
test_LoggedOutInfo .................................... 255
test_InvalidRequestInfo ................................ 256
test_isCorrectUsername ................................. 256
test_isCorrectPassword ................................. 256

Detailed documentation for all declarations in this file follows.

7.22.1 LoginTestCase

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 38:
class LoginTestCase(unittest.TestCase):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase

Description  The tests in this script are created conform the API of /query/schedule/authenticate.py and /query/schedule/load.py. That means that the asserts test on the results these files would give according to their API.

7.22.2 LoginTestCase.denyAllLogins

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 51:
def denyAllLogins(self, msg_identifier):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.denyAllLogins
7.22. KROKETAPP.TESTS.QUERY.USER.TESTAUTHENTICATE

**Description**  Help method for login tests. All login requests should be rejected, since neither of these accounts are registered.

### 7.22.3 LoginTestCase.assertValidJSON_and_Load

**Declaration**  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 67:
```
def assertValidJSON_and_Load(self, response, msg_identifier):
```

**Qualified name**  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.assertValidJSON_and_Load

**Description**  Help method for login tests. Checks if the response of the server is in JSON format. If it is, then parse the response and return that. Else return nothing and the test fails.

### 7.22.4 LoginTestCase.test_ValidLogin

**Declaration**  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 91:
```
def test_ValidLogin(self):
```

**Qualified name**  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_ValidLogin

**Description**  Test for logging in with valid accounts. A login query is valid if it contains an username and a password, and it is an AJAX and POST request. It tests the following:

- Valid login query, but not existing account. Should return 200.
- Invalid login query, it has no username. Should return 403.
- Invalid login query, it has no password. Should return 403.
- Invalid login query, it has no username nor password. Should be return 403.
- Invalid login query, it is no POST request. Should be return 403.
- Invalid login query, it is no AJAX request. Should be return 403.
- Valid login query. Should be true.
- Valid login query, but not existing account (anymore). Should be 200.
- Valid info query. Account is not logged in anymore. Should be 200.

### 7.22.5 LoginTestCase.test_InvalidLoginTest

**Declaration**  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 171:
```
def test_InvalidLoginTest(self):
```

251
CHAPTER 7. UNIT TESTS

Qualified name KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_InvalidLoginTest

Description Tests for logging in with not valid accounts. A login query is valid if it contains an username and a password, and it is an AJAX and POST request. It tests the following:

- Valid login query, but not existing account. Should return 200.
- Invalid login query, it has no username. Should be return 403.
- Invalid login query, it has no password. Should be return 403.
- Invalid login query, it has no username nor password. Should be return 403.
- Invalid login query, it is no POST request. Should be return 403.
- Invalid login query, it is no AJAX request. Should be return 403.

7.22.6 LoginTestCase(assertValidJSON_and_Load)

Declaration Declared in KroketApp/tests/query/user/testAuthenticate.py on line 215:
def assertValidJSON_and_Load(self, response, msg_identifier):

Qualified name KroketApp.tests.query.user.testAuthenticate.LoginTestCase.assertValidJSON_and_Load

Description Help method for logout tests. Checks if the response of the server is in JSON format. If it is, then parse the response and return that. Else return nothing and the test fails.

7.22.7 LoginTestCase.test_InvalidPost

Declaration Declared in KroketApp/tests/query/user/testAuthenticate.py on line 244:
def test_InvalidPost(self):

Qualified name KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_InvalidPost

Description Tests for logging out with valid accounts. A logout query is valid if the user is logged in (tested with session) and it is an AJAX and POST request. It tests the following:

- Valid logout query, but the user is not logged in. Should return 200.
- Invalid logout query, it is no AJAX request. Should return 403.
- Valid info query. Account should still be logged in due invalid log out query. Should be true.
7.22. KROKETAPP.TESTS.QUERY.USER.TESTAUTHENTICATE

- Invalid logout query, it is no POST request. Should return 403.
- Valid info query. Account should still be logged in due invalid log out query. Should be true.
- Invalid logout query, it is no POST nor AJAX request. Should return 403.
- Valid info query. Account should still be logged in due invalid log out query. Should be true.
- Valid logout query. Account was logged in. Should return 200.
- Valid info query. Account should be logged out. Should be false.

7.22.8 LoginTestCase.test_BasicLogouts

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 313:
def test_BasicLogouts(self):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_BasicLogouts

Description  Tests for logging out with valid accounts. A logout query is valid if the user is logged in (tested with session) and it is an AJAX and POST request. It tests the following:

- Valid logout query. Account was logged in. Should return 200.
- Valid info query. Account should be logged out. Should be false.

7.22.9 LoginTestCase.assertValidJSON_and_Load

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 347:
def assertValidJSON_and_Load(self, response, msg_identifier):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.assertValidJSON_and_Load

Description  Help method for registering tests. Checks if the response of the server is in JSON format. If it is, then parse the response and return that. Else return nothing and the test fails.

7.22.10 LoginTestCase.test_ValidCredentialsRegisters

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 364:
def test_ValidCredentialsRegisters(self):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_ValidCredentialsRegisters
CHAPTER 7. UNIT TESTS

Description  Tests for registering valid accounts. A register query is valid if the username is unique, both username and password are valid, and it is an AJAX and POST request. It tests the following:

- Valid register query. Account is registered. Should return 200.
- Valid login query. Account should be logged in. Should return 200.
- Valid register query, but account name already exists. Should return 200.

7.22.11  LoginTestCase.test_WrongUsernamePasswordRegisters

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 401:
def test_WrongUsernamePasswordRegisters(self):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_WrongUsernamePasswordRegisters

Description  Tests for registering invalid accounts. A register query is valid if the username is unique, both username and password are valid, and it is an AJAX and POST request. It tests the following:

- Valid register query. Account is registered. Should return 200.
- Valid login query. Account should be logged in. Should return 200.
- Valid register query, but account name already exists. Should return 200.

7.22.12  LoginTestCase.test_InvalidRequests

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 431:
def test_InvalidRequests(self):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_InvalidRequests

Description  Tests for registering (in)valid accounts. A register query is valid if the username is unique, both username and password are valid, and it is an AJAX and POST request. It tests the following:

- Invalid register query, it has no username. Should be return 403.
- Invalid register query, it has no password. Should be return 403.
- Invalid register query, it has no username nor password. Should be return 403.
- Invalid register query, it is no AJAX request. Should be return 403.
- Invalid register query, it has no username nor AJAX request. Should be return 403.
• Invalid register query, it has no password nor AJAX request. Should be return 403.

• Invalid register query, it has no username nor password nor AJAX request. Should be return 403.

7.22.13 LoginTestCase.assertValidJSON_and_Load

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 472:
def assertValidJSON_and_Load(self, response, msg_identifier):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.assertValidJSON_and_Load

Description  Help method for info tests. Checks if the response of the server is in JSON format. If it is, then parse the response and return that. Else return nothing and the test fails.

7.22.14 LoginTestCase.test_LoggedInInfo

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 487:
def test_LoggedInInfo(self):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_LoggedInInfo

Description  Tests for info about accounts. A info query is valid if it is an AJAX and POST request. It tests the following:

• Valid info query. Account is registered and logged in. Should return true.
• Valid info query. Account is registered and logged out. Should return true.

7.22.15 LoginTestCase.test_LoggedOutInfo

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 518:
def test_LoggedOutInfo(self):

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_LoggedOutInfo

Description  Tests for info about accounts. A info query is valid if it is an AJAX and POST request. It tests the following:

• Valid info query. Account is not registered. Should return false.
• Valid info query. Tried to login, but account is not registered. Should return false.
CHAPTER 7. UNIT TESTS

7.22.16 LoginTestCase.test_InvalidRequestInfo

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 546:

```python
def test_InvalidRequestInfo(self):
```

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_InvalidRequestInfo

Description  Tests for info about accounts. A info query is valid if it is an AJAX and POST request. It tests the following:

- Invalid info query, is not AJAX request and not logged in. Should return 403.
- Invalid info query, is not AJAX request. Should return 403.

7.22.17 LoginTestCase.test_isCorrectUsername

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 572:

```python
def test_isCorrectUsername(self):
```

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_isCorrectUsername

Description  Tests for validating username. If username meets the requirements, then the result is true. Else false.

7.22.18 LoginTestCase.test_isCorrectPassword

Declaration  Declared in KroketApp/tests/query/user/testAuthenticate.py on line 582:

```python
def test_isCorrectPassword(self):
```

Qualified name  KroketApp.tests.query.user.testAuthenticate.LoginTestCase.test_isCorrectPassword

Description  Tests for validating password. If password meets the requirements, then the result is true. Else false.
7.23. KROKETAPP.TESTS.QUERY.USER.TESTCHANGE

7.23 KroketApp.tests.query.user.testChange

A Python script that tests the methods of /query/user/change.py.

Hierarchical member index  This file contains the following members:

ChangePropertiesTestCase ............................................. 257
assertValidJSON_and_Load ............................................. 257
test_successEmptyQuery ............................................. 257
test_negSuccessEmptyQuery ........................................... 258
test_successPasswordQuery ............................................ 258
def test_negSuccessPasswordQuery ................................. 258
test_invalidQuery ..................................................... 259

Detailed documentation for all declarations in this file follows.

7.23.1 ChangePropertiesTestCase

Declaration  Declared in KroketApp/tests/query/user/testChange.py on line 34:
class ChangePropertiesTestCase(unittest.TestCase):

Qualified name  KroketApp.tests.query.user.testChange.ChangePropertiesTestCase

Description  The tests in this script are created conform the API of /query/schedule/authenticate.py
and /query/schedule/load.py. That means that the asserts test on the results these files
would give according to their API.

7.23.2 ChangePropertiesTestCase.assertValidJSON_and_Load

Declaration  Declared in KroketApp/tests/query/user/testChange.py on line 48:
def assertValidJSON_and_Load(self,response, msg_identifier):

Qualified name  KroketApp.tests.query.user.testChange.ChangePropertiesTestCase.
assertValidJSON_and_Load

Description  Help method for change password tests. Checks if the response of the server
is in JSON format. If it is, then parse the response and return that. Else return nothing and
the test fails.

7.23.3 ChangePropertiesTestCase.test_successEmptyQuery

Declaration  Declared in KroketApp/tests/query/user/testChange.py on line 66:
def test_successEmptyQuery(self):
CHAPTER 7. UNIT TESTS

Qualified name KroketApp.tests.query.user.testChange.ChangePropertiesTestCase.test_successEmptyQuery

Description Test for changing passwords with no new password. A change query is valid if it contains the current password and a new password, both passwords are valid, and it is an AJAX and POST request. It tests the following:

- Valid change query, but has no new password. Should return 200.
- Valid login query. Account should still be able to log in with ‘old’ password.

7.23.4 ChangePropertiesTestCase.test_negSuccessEmptyQuery

Declaration Declared in KroketApp/tests/query/user/testChange.py on line 98:

def test_negSuccessEmptyQuery(self):

Qualified name KroketApp.tests.query.user.testChange.ChangePropertiesTestCase.test_negSuccessEmptyQuery

Description Test for changing passwords with no new password. A change query is valid if it contains the current password and a new password, both passwords are valid, and it is an AJAX and POST request. It tests the following:

- Valid change query, but has no new password and current password is false. Should return 200.

7.23.5 ChangePropertiesTestCase.test_successPasswordQuery

Declaration Declared in KroketApp/tests/query/user/testChange.py on line 128:

def test_successPasswordQuery(self):

Qualified name KroketApp.tests.query.user.testChange.ChangePropertiesTestCase.test_successPasswordQuery

Description Test for changing passwords. A change query is valid if it contains the current password and a new password, both passwords are valid, and it is an AJAX and POST request. It tests the following:

- Valid change query. Should return 200.
- Valid info query. Account should still be logged in. Should be true.
- Valid info query. Account should be able to log in with new account. Should be true.

7.23.6 ChangePropertiesTestCase.def test_negSuccesPasswordQuery

Declaration Declared in KroketApp/tests/query/user/testChange.py on line 169:

def test_negSuccesPasswordQuery(self):
Qualified name KroketApp.tests.query.user.testChange.ChangePropertiesTestCase.
def test_negSuccesPasswordQuery

Description Test for changing passwords. A change query is valid if it contains the current password and a new password, both passwords are valid, and it is an AJAX and POST request. It tests the following:

- Invalid change query, new password is not valid. Should return 200.
- Valid info query. Account should still be able to login with old password. Should be true.
- Valid change query, but current password is false. Should return 200.
- Valid info query. Account should still be able to login with old password. Should be true.

7.23.7 ChangePropertiesTestCase.test_invalidQuery

Declaration Declared in KroketApp/tests/query/user/testChange.py on line 230:
def test_invalidQuery(self):

Qualified name KroketApp.tests.query.user.testChange.ChangePropertiesTestCase.
test_invalidQuery

Description Test for changing passwords. A change query is valid if it contains the current password and a new password, both passwords are valid, and it is an AJAX and POST request. It tests the following:

- Invalid change query, account is not logged in. Should return 403.
- Invalid change query, no current password. Should return 403.
- Invalid change query, is not AJAX request. Should return 403.
- Invalid change query, is not AJAX request and has no current password. Should return 403.
- Invalid change query, account is not logged in and wrong password. Should return 403.
- Invalid change query, account is not logged in and has no current password. Should return 403.
- Invalid change query, account is not logged in. Should return 403.
Chapter 8

Index

In this chapter, you will find a list of all identifiers used in all of the kroket code (except for the HTML code). Along with every identifier, the page number (in this document) of its description is given. Furthermore the programming language and type of the definition is indicated using an abbreviation:

- **css**: a CSS rule;
- **js fun**: a JavaScript function;
- **js var**: a JavaScript variable;
- **py cls**: a Python class;
- **py met**: a Python method;
- **py var**: a Python variable.

**Symbols**

- _: **js var**, 68, 69
- __init__: **py met**, 109, 183
- __unicode__: **py met**, 87, 88, 91, 94, 96, 98, 100, 101, 120, 121, 123, 124
- _call_: **py met**, 184
- _getClient_: **py met**, 183

A

- a: **css**, 27
- Account: **py cls**, 117
- account: **py var**, 122
- AccountAdmin: **py cls**, 106
- .addCourse: **css**, 20
- addCourse: **py met**, 177
- addCoursePlanning: **py met**, 177
- addDeleteIcon(course): **js fun**, 75
- addSoftwareScienceMajor: **py met**, 178
addYear(): js fun, 35
>alertModal: css, 28
>allowed: py var, 112
>ALLOWED_TYPES: py var, 135
>AMOUNT_OF_RESULTS: py var, 140
>askLoadName(success): js fun, 57
>askMajorCourses(): js fun, 36
>askSaveName(success): js fun, 57
>assertValidJSON and Load: py met, 225, 239, 241, 246, 251–253, 255, 257
>autoHiding(): js fun, 48

B

basicTest(): js fun, 191
>begin_year: py var, 123
>biztalk: py var, 175
>BIZTALK_RETURN_TYPEDEF: py var, 181
>buildTable(years): js fun, 46

C

changePassword: py met, 119, 173
>changeProperties: py met, 172
>ChangePropertiesTestCase: py cls, 257
>CHAR_SET: py var, 117
>check_credentials: py met, 129
>check_loggedIn: py met, 130
>check_loggedOut: py met, 131
>check_password: py met, 120
>check_validQuery: py met, 130
>checkCoherent: py met, 155
>checkCoherent(data): js fun, 62
>checkDoubleCourses(cell): js fun, 39
>checkDoubleCoursesRecursive(courses, index): js fun, 76
>checkEcts(courses): js fun, 63
>checkLoggedIn(showStart): js fun, 50
>checkMajor(data): js fun, 62
>checkOverlap: py met, 155
>checkOverlap(data): js fun, 61
>checkSavenameAndSave(savename): js fun, 56
>checkScheduleOverwrite: py met, 147
>checkUSE: py met, 154
>checkUse(data): js fun, 62
>clearData: py met, 224
>clearDB: py met, 176
CHAPTER 8. INDEX

clickedCell(event, cell): js fun, 38
clickedCoherentPackage(event, pack): js fun, 37
clickedCourse(course): js fun, 38
closeFiles: py met, 176
.closeWhite: css, 23
.closeWhite:hover: css, 23
code: py var, 82, 121
COHERENT: py var, 90
CoherentPackage(target_group, name): js fun, 37
coherentSearchButtonWaiting(bool): js fun, 44
CoherentTestCase: py cls, 228
colorCourses(): js fun, 72
commitSession: py met, 131
.course.major: css, 21
content: py var, 83
CONTENT_TYPE: py var, 223, 235, 242
convert: py met, 158
ConvertTestCase: py cls, 240
count: py var, 99, 101
countEcts(courses): js fun, 63
Course: py var, 185
.course: css, 20, 29
.course > div:not(.close): css, 21
.course.disabledCourse: css, 21
CourseInfo(code, name, ects, level, lastYear, planning, priorknowledge): js fun, 36
.course.major: css, 21, 29
CoursePlanning: py cls, 93
.course.selectedCourse: css, 20
.course.selectedCourse.major: css, 21
CPInline: py cls, 104
createTuples: py met, 188
currentScheduleName: js var, 34

D
decodeJSON: py met, 147
def getPriorKnowledge: py met, 233
def test_negSuccesPasswordQuery: py met, 258
deleteDoubles: py met, 189
deleteOldTable: py met, 189
deleteSchedule: py met, 151
denyAllLogins: py met, 250
department: py var, 85
disableCourses(): js fun, 73
doDragging(): js fun, 47
doDropping(): js fun, 47
doLogout(): js fun, 50
.dropdown-menu .disabledMenuItem: css, 18
dropdown-menu .disabledMenuItem i: css, 18
dropdown-menu .disabledMenuItem:hover: css, 18
dutch_stopwords: py var, 111

E
ects: py var, 82
education_type: py var, 85
emptyPasswords(): js fun, 52
emptyPersistentData(): js fun, 59
emptyResponse: py met, 143
emptySchedule(): js fun, 40
end: js var, 49
english_stopwords: py var, 111
equal_lists: py met, 236, 243
equal_subject: py met, 237, 244
evaluation: css, 28
examination_type: py var, 85
expec: py met, 237, 244
EXPIRE SECONDS: js var, 58
extendedFormat: py met, 161

F
Failure: py var, 184
FALSE: py var, 149, 167, 172
FIELDS: py var, 238, 245
fillInCourses(courses, major): js fun, 35
fillInSchedule(): js fun, 35
filterPackageName(name): js fun, 73
filterTimeSlot(slot): js fun, 73
first: js var, 39
fixTabletMenus(): js fun, 48
follow_up: py met, 160
follow_up: py var, 96, 238, 245
fuzzySubjectSearch: py met, 137, 165

G
geefVakGegevens: py met, 186
CHAPTER 8. INDEX

generateResponse: py met, 142
get_data: py met, 109
getAdditionalInfo: py met, 178
getClone(course): js fun, 78
getFollowUp: py met, 87, 233
getList: py met, 133
getMajorName: py met, 143
getMajors(func): js fun, 47
getMandatorySubjects: py met, 92
getNrOfDoubles: py met, 189
getPersistentData(): js fun, 60
getPlanning: py met, 95, 159
getPlannings: py met, 234
getPostDataSearch(): js fun, 41
getPrintVersion(): js fun, 49
getPriorKnowledge: py met, 87
getRelatedSubjects: py met, 128
getResultAttr: py met, 182
getResultList: py met, 182
getSavedList(success): js fun, 55
getSchedule: py met, 142
getSchedule(onlyElective): js fun, 74
getSubjectPackage: py met, 233
getSubjects: py met, 92
getTestSet: py met, 232

H
handle_data: py met, 109
HASH_TIMES: py var, 118
hashPassword: py met, 119
.hiddenSubmit: css, 28
hiddenTab: js var, 73
hideDisabler(): js fun, 71
hideStartModal(): js fun, 72
.highlight: css, 19
.highlight > .addCourse: css, 20
highlight(course): js fun, 75
.highlight:hover: css, 19
.hoveredHighlight: css, 19
html_tags: py var, 212

I
id: py var, 93, 96, 97, 99, 101, 122

264
info: py met, 157, 169
InfoTestCase: py cls, 238
inPackage: py met, 156
is_ascii: py met, 170
isCorrectName: py met, 146
isCorrectPassword: py met, 170
isCorrectUsername: py met, 170
isDragging: js var, 32
isEq(a1, a2): js fun, 201
isLoggedIn: js var, 33
isPersistentDataSupported(): js fun, 60
isPersistentSupported: js var, 60
isValidPassword(password): js fun, 53
isValidUsername(username): js fun, 52

K
Keyword: py cls, 88
keyword: py var, 99, 101
.keyword-highlighting: css, 24
keyword_test_set: py var, 211
KeywordAdmin: py cls, 106
keywordHighlight(str): js fun, 43
KeywordPacketRelation: py cls, 100
KeywordSubjectRelation: py cls, 98
KWPInline: py cls, 104
KWSInline: py cls, 104

L
.label: css, 21
LANGUAGE_NL: py var, 184
last_year: py var, 84
lastSearchTerm: js var, 41
leesvak: py met, 187
.less-vertical-space: css, 24
lmtzr: py var, 110, 135, 163
load(): js fun, 46
loadClick(): js fun, 54
loadData: py met, 224
loadPersistentData(): js fun, 59
loadSchedule: py met, 141
loadScheduleFromServer(savename, bool): js fun, 55
lockAuthentication: js var, 32
login: py met, 168
CHAPTER 8. INDEX

login(): js fun, 51
login2(): js fun, 51
loginGUI(username): js fun, 51
LoginTestCase: py cls, 250
loginToServer(username, password, bool, load): js fun, 51
logout: py met, 169
logout(): js fun, 50

M
main: py met, 179, 189
MAJOR: py var, 89
major: js var, 33
major: py var, 122, 128
majorCourses: js var, 34
majorName: js var, 34
makePackage(data): js fun, 76
makeSubject(data): js fun, 77
mandatory: py var, 94
Meta: py cls, 92, 95, 97, 98, 100, 102, 123, 125, 128
MIN_COHERENT: py var, 152
MIN_USE: py var, 153
MLStripper: py cls, 108
modifiedSchedule: js var, 32
multiplier: py var, 112

N
name: py var, 82, 88, 90, 118, 122
.navbar .brand: css, 27
.navbar .nav > .active > a, .navbar .nav > .active > a:hover, .navbar .nav >
    .active > a:focus: css, 27
.navbar .nav > li > a, .navbar .brand, .navbar .nav > li > a:focus, .navbar
    .nav > li > a:hover: css, 25
.navbar .nav > li > a:hover: css, 25
.navbar .nav li.dropdown.open > .dropdown-toggle, .navbar .nav li.dropdown.active
    > .dropdown-toggle, .navbar .nav li.dropdown.open.active > .dropdown-toggle:
    css, 27
.navbar-fixed-top .navbar-inner, .navbar-static-top .navbar-inner: css, 27
.navbar-inner: css, 25
niveau: py var, 82
noConflict(courses): js fun, 77
noCurrConflict(c1, c2): js fun, 78
NON_STANDARD_FORMAT_FIELDS: py var, 86, 91

O
objects: py var, 86, 91
.onBottom: css, 22
openFiles: py met, 175
OverlapTestCase: py cls, 227
owisCourses: py met, 178

P
.package: css, 21
.package:last-child: css, 22
packageTypeToString(number): js fun, 70
parseDoelgroep: py met, 176
parseProgramma: py met, 177
password: py var, 118
PatchWSDLPlugin: py cls, 181
planning: py var, 237, 238, 244, 245
.popover-error .popover-title: css, 22
prior_knowledge: py met, 159
prior_knowledge: py var, 98, 236, 239, 243, 246

Q
quartile: py var, 94

R
rankDoelgroep: py met, 179
rankEnglish: py met, 114
rankPacket: py met, 114
rankStringNLTK: py met, 113
rankSubject: py met, 112
rating: py var, 86, 128
receiveMajors(data): js fun, 47
recommend_timeout: js var, 40
.recommended: css, 23
#recommended-subjects-container .keyword-highlighting: css, 24
recommendSubjects(): js fun, 40
register: py met, 119, 168
register(): js fun, 52
RelatedSubject: py cls, 127
RelatedSubjectAdmin: py cls, 107
remarks: py var, 83, 90
removeClick(): js fun, 55
removeCourse(clickedIcon): js fun, 38
removeHighlight(): js fun, 75
removeKeywordHighlight(): js fun, 42
CHAPTER 8. INDEX

removeSchedule: py met, 151
removeYear(): js fun, 34
renameScheduleName: py met, 149
resetTableRows(): js fun, 62
returnSubject: py met, 146

S
salt: py var, 118
SALT_LENGTH: py var, 117
saveAsClick(): js fun, 54
saveClick(): js fun, 54
saveMajor(): js fun, 36
saveschedule: py met, 145
saveScheduleToDb: py met, 147
saveScheduleToServer(savename): js fun, 56
Schedule: py cls, 121
schedule: py var, 124
#schedule-container .table: css, 17
ScheduleAdmin: py cls, 106
ScheduleInline: py cls, 105
ScheduleSubject: py cls, 120
ScheduleSubjectAdmin: py cls, 106
ScheduleSubjectRelation: py cls, 123
ScheduleSubjectRelInline: py cls, 105
search(): js fun, 42
#searchbar .nav-tabs: css, 24
#searchbar .tab-content: css, 24
searchButtonWaiting(bool): js fun, 42
searchCoherentPackage(): js fun, 44
searchmajor: py met, 134
searchpackage: py met, 136, 139
SearchTestCase: py cls, 245
selectedCourse: js var, 33
SESSION_EXPIRING: py var, 129
setCourses(pack): js fun, 38
setExpanded(expanded): js fun, 73
setMajor(m, mName): js fun, 71
setModified(bool): js fun, 71
setRecommendedSubjects(subjects): js fun, 40
setSubjectInfo(info): js fun, 74
setUp: py met, 208, 212–214, 224, 227, 229, 230, 239, 240, 246
setYear(y): js fun, 34

268
CHAPTER 8. INDEX

.target_group: py var, 90, 93, 101
tearDown: py met, 212–214, 225, 228–230, 239, 240, 246
tempDB: py var, 188
test_AddCourse: py met, 205
test_AddCoursePlanning: py met, 206
test_AddSoftwareScienceMajor: py met, 207
test_all_schedules: py met, 221
test_BasicLogouts: py met, 253
test_BasicSearchCodeTerm: py met, 247
test_ClearDB: py met, 205
test_CreateTuples: py met, 208
test_credentials: py met, 216
test_delete_schedule: py met, 221
test_DeleteDoubles: py met, 209
test_DeleteOldTable: py met, 209
test_difficulty: py met, 248
test_fail: py met, 228–230
test_failCoherent: py met, 226
test_failMajor: py met, 227
test_failOverlap: py met, 226
test_failUSE: py met, 226
test_GetAditionalInfo: py met, 206
test_GetNrOfDoubles: py met, 209
test_InvalidLoginTest: py met, 251
test_InvalidPost: py met, 252
test_invalidQuery: py met, 259
test_invalidRequest: py met, 225, 239, 241, 247
test_InvalidRequestInfo: py met, 256
test_InvalidRequests: py met, 254
test_isCorrectPassword: py met, 256
test_isCorrectUsername: py met, 256
test_list: py met, 217
test_load_non_existing_schedule: py met, 220
test_load_non_related_schedule: py met, 220
test_load_schedule: py met, 220
test_LoggedInInfo: py met, 255
test_LoggedOutInfo: py met, 255
test_negSuccessEmptyQuery: py met, 258
test_overwrite_schedule: py met, 221
test_OwisCourses: py met, 206
test_package: py met, 217, 218
test_ParseDoelgroep: py met, 205
test_ParseProgramma: py met, 206
test_pass: py met, 228–230
test_passAll: py met, 225
test_quartile: py met, 248
test_range: py met, 247
test_rankEnglish: py met, 213
test_rankPacket: py met, 215
test_rankStringNLTK: py met, 212
test_recommendations: py met, 219
test_rename_schedule: py met, 221
test_search: py met, 218
test_SearchTermFormat: py met, 247
test_successEmptyQuery: py met, 257
test_successPasswordQuery: py met, 258
test_targetGroup: py met, 249
test_timeSlot: py met, 248
test_updateSubDB: py met, 214
test_ValidCredentialsRegisters: py met, 253
test_ValidLogin: py met, 251
test_validRequest: py met, 240, 241
test_WrongUsernamePasswordRegisters: py met, 254
test_year: py met, 248
testAddData: py cls, 205
testAddRemoveYear(): js fun, 192
testAskingMajors(): js fun, 195
testAskingMajorsMocking(): js fun, 195
testBuildTable(): js fun, 195
testCheckCoherent(): js fun, 198
testCheckDoubleCourses(): js fun, 193
testCheckDoubleCoursesRecursive(): js fun, 200
testCheckMajor(): js fun, 198
testCheckOverlap(): js fun, 198
testCheckUse(): js fun, 198
testCoherentSearchButtonWaiting(): js fun, 204
CHAPTER 8. INDEX

testColorCourses(): js fun, 193
testDisableCourses(): js fun, 199
testEmptySchedule(): js fun, 193
testFillInCourses(): js fun, 192
testFilterTimeSlot(): js fun, 200
testGetPostdataSearch(): js fun, 203
testGetSchedule(): js fun, 200
testHighlight(): js fun, 200
testKeywordHighlight(): js fun, 203
testMakePackage(): js fun, 201
testMakeSubject(): js fun, 201
testNoConflict(): js fun, 201
testNoCurrConflict(): js fun, 201
testPacket: py cls, 214
testRankStringNLTK: py cls, 212
testRecommender_job: py cls, 208
testRemoveCourse(): js fun, 193
testRemoveHighlight(): js fun, 200
testRemoveKeywordHighlight(): js fun, 203
testSaveMajor(): js fun, 193
testSaveName(): js fun, 202
testSetModified(): js fun, 199
testSetRecommendedSubjects(): js fun, 194
testSetScheduleLoading(): js fun, 199
testSetSubjectInfo(): js fun, 200
testSetYear(): js fun, 192
testShowCoherentPackages(): js fun, 204
testShowSchedule(): js fun, 202
testShowSearchResults(): js fun, 203
testShowSubjectInformation(): js fun, 193
testSubject: py cls, 213
testUpdateMajorErrorBar(): js fun, 192
testValidPassword(): js fun, 196
testValidUsername(): js fun, 196
tg: py met, 233
time_block: py var, 124
timeslot: py var, 94
toStandardFormat: py met, 158
total_count: py var, 88
translateQStoJSON: py met, 143
TRUE: py var, 149, 167, 172
TueBizTalk: py cls, 183
TueBizTalkException: py cls, 181
TueBizTalkInternalError: py cls, 181
TueBizTalkMissingAttribute: py cls, 182

type: py var, 91
TYPES: py var, 90

U

unsorted_equal: py met, 236, 243
updateMajorErrorBar(bool): js fun, 35
updatePacketDB: py met, 115
updatePersistentData(): js fun, 59
updateSubDB: py met, 113
url: py var, 85
url_converter: js var, 67
url_elPackageList: js var, 64
url_isLoggedIn: js var, 66
url_load: js var, 66
url_login: js var, 67
url_logout: js var, 67
url_majorList: js var, 64
url_majorSubjects: js var, 65
url_packageSearch: js var, 65
url_packageSubjects: js var, 65
url_recommend: js var, 66
url_register: js var, 67
url_removeSchedule: js var, 66
url_save: js var, 66
url_savedList: js var, 66
url_subjectInfo: js var, 65
url_subjectSearch: js var, 65
url_usePackageList: js var, 65
url_validate: js var, 67
url_patterns: py var, 126
USE: py var, 89
USERNAME_FORBIDDEN_SYMBOLS: py var, 169
userschedules: py met, 148
USETestCase: py cls, 230

V

VakVervolgVak: py var, 185
VakVoorkennis: py var, 185
validate: py met, 153
CHAPTER 8. INDEX

validateSchedule(): js fun, 61
ValidateTestCase: py cls, 224
VerzorgendeEenheid: py var, 185
video_mat_descr: py var, 84
video_url: py var, 84
view: py met, 140
VQSToDict: py met, 138, 166

W
warning(str, cell): js fun, 39
weekly_content: py var, 83
wordnet_dict: py var, 112

X
XSD_PATH: py var, 180

Y
year: py var, 94
years: js var, 33