Enwticklerdokumentation: Antragsverwaltungstool (I3)

Inhaltsverzeichnis

Ziel der Dokumentation

Ziel der Dokumentation ist es den künftigen Entwicklern der Software möglichst viel Infors bereitzustellen. Die Dokumentation is bei Entwicklung entsprechend zu ergänzen.

Entwurfsdokumentation

Systemarchitektur

logische Sicht

```
@startuml "logische Sicht"

node "Django-Applikation" {
    [HTML]
    [View]
        note left of [View] : CRUD, Processes requests
    [CSS]
        note left of [CSS]
            predefined W3
            and own stylesheet
        end note
    [URL Resolver]
        note left of [URL Resolver] : recieves request by url paths

    [JavaScript] ..> [HTML] : toggles, displays text
    [CSS] ..> [HTML] : styles
    [URL Resolver] --> [View] : redirects request
    [View] ..> [HTML] : renders response to browser

    [View] --> database : CRUD-Operations
}

database "MySQL-Datenbank" as database {

}
@enduml
```

Bibliotheken und Frameworks

Frameworks

Als Framework wird Django verwendet. Django erleichtert ganz klar die Arbeit in der Applikation, da für die Kommunikation alle Funktionalitäten bereitgestellt werden. Außerdem lassen sich über das MVC-Prinzip alle Funktionen gut abstrahieren und einzeln bearbeiten.

Bibliotheken

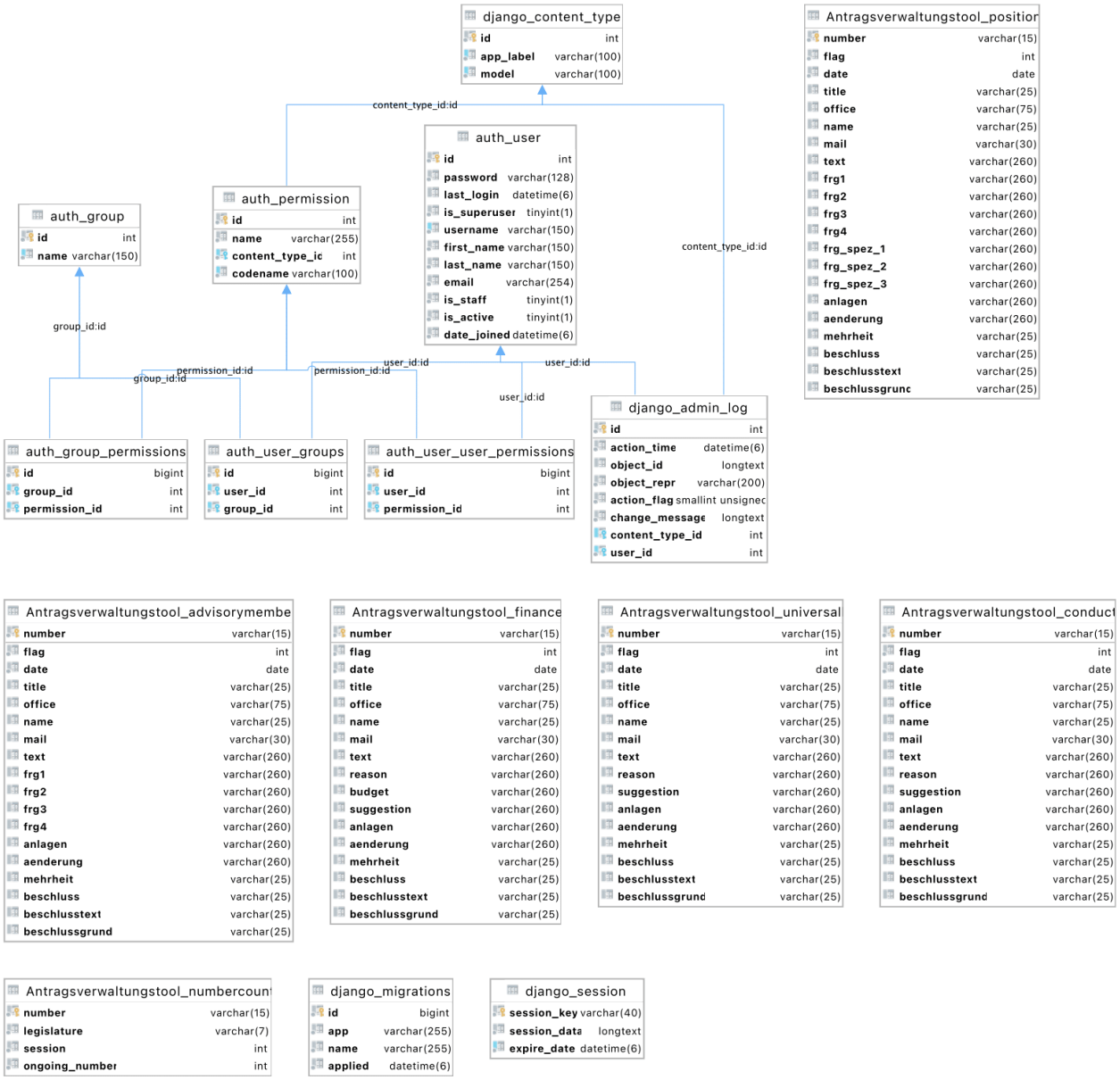
Es werden ansonsten nur einzelne Django-Bibliotheken genutzt. Nebenbeiwird noch ein Stylesheet von W3-Schools genutzt.

Datenbankschema

Das Datenbankschema wird von Django verwaltet. Nach Änderungen in den einzelnen Modellen kann die Änderung in die Datenbank migriert werden. Hier gibt es eine Tabelle mit den Login-Daten (*auth_user*) der eigens angelegten Nutzer der Applikation. Alle anderen django-spezifischen Tabellen müssen nicht eigens verwaltet werden. Hier befinden sich Zugriffsrechte und der

Adminlogin, sowie die einzelnen Migrationen der Modelle in die Datenbank.

Die wichtigsten Tabelle sind die Tabellen die nach den in Django festgelegten Modellen migriert(erstellt/geändert) werden. Hier finden sich die Tabellen: *Antragsverwaltungstool_advisorymember*, *Antragsverwaltungstool_finance*, *Antragsverwaltungstool_position*, *Antragsverwaltungstool_universall*. Die Spalten in den Tabellen spiegeln jeweils Attribute eines Elements erstellt nach dem Model wieder. Es ist wichtig zu beachten das sich die Tabelle je nach den Models ändern und dadurch je nach Erweiterung "dynamisch" werden !


django_content_type
id int
app_label varchar(100)
model varchar(100)
content_type_id:id
auth_user
id int
password varchar(128)
last_login datetime(6)
is_superuser tinyint(1)
username varchar(150)
first_name varchar(150)
last_name varchar(150)
email varchar(254)
is_staff tinyint(1)
is_active tinyint(1)
date_joined datetime(6)
Antragsverwaltungstool_position
number varchar(15)
flag int
date date
title varchar(25)
office varchar(75)
name varchar(25)
mail varchar(30)
text varchar(260)
frg1 varchar(260)
frg2 varchar(260)
frg3 varchar(260)
frg4 varchar(260)
frg_spez_1 varchar(260)
frg_spez_2 varchar(260)
frg_spez_3 varchar(260)
anlagen varchar(260)
aenderung varchar(260)
mehrheit varchar(25)
beschluss varchar(25)
beschlusstext varchar(25)
beschlussgrunc varchar(25)
auth_group
id int
name varchar(150)
auth_permission
id int
name varchar(255)
content_type_ic int
codename varchar(100)
content_type_id:id
group_id:id
permission_id:id
group_id:id
permission_id:id
user_id:id
user_id:id
user_id:id
auth_group_permissions
id bigint
group_id int
permission_id int
auth_user_groups
id bigint
user_id int
group_id int
auth_user_user_permissions
id bigint
user_id int
permission_id int
django_admin_log
id int
action_time datetime(6)
object_id longtext
object_repr varchar(200)
action_flag smallint unsignec
change_message longtext
content_type_id int
user_id int
Antragsverwaltungstool_advisorymembe
number varchar(15)
flag int
date date
title varchar(25)
office varchar(75)
name varchar(25)
mail varchar(30)
text varchar(260)
frg1 varchar(260)
frg2 varchar(260)
frg3 varchar(260)
frg4 varchar(260)
anlagen varchar(260)
aenderung varchar(260)
mehrheit varchar(25)
beschluss varchar(25)
beschlusstext varchar(25)
beschlussgrund varchar(25)
Antragsverwaltungstool_finance
number varchar(15)
flag int
date date
title varchar(25)
office varchar(75)
name varchar(25)
mail varchar(30)
text varchar(260)
reason varchar(260)
budget varchar(260)
suggestion varchar(260)
anlagen varchar(260)
aenderung varchar(260)
mehrheit varchar(25)
beschluss varchar(25)
beschlusstext varchar(25)
beschlussgrund varchar(25)
Antragsverwaltungstool_universal
number varchar(15)
flag int
date date
title varchar(25)
office varchar(75)
name varchar(25)
mail varchar(30)
text varchar(260)
reason varchar(260)
suggestion varchar(260)
anlagen varchar(260)
aenderung varchar(260)
mehrheit varchar(25)
beschluss varchar(25)
beschlusstext varchar(25)
beschlussgrund varchar(25)
Antragsverwaltungstool_conduct
number varchar(15)
flag int
date date
title varchar(25)
office varchar(75)
name varchar(25)
mail varchar(30)
text varchar(260)
reason varchar(260)
suggestion varchar(260)
anlagen varchar(260)
aenderung varchar(260)
mehrheit varchar(25)
beschluss varchar(25)
beschlusstext varchar(25)
beschlussgrund varchar(25)
Antragsverwaltungstool_numbercoun
number varchar(15)
legislature varchar(7)
session int
ongoing_number int
django_migrations
id bigint
app varchar(255)
name varchar(255)
applied datetime(6)
django_session
session_key varchar(40)
session_data longtext
expire_date datetime(6)

Zusammenarbeit zwischen den Komponenten

Anzeigen

Ablauf eines Aufrufs zum Anzeigen

```
@startuml "Anzeigen"

Browser -> Template : request
Template -> View : request with user input
View -> Model : get data according to user input
Model -> View : response data
View -> Template : render with data
Template -> Browser : content

@enduml
```

Daten einfügen

Ablauf eines Aufrufs zum Einfügen

```
@startuml "Einfügen"

Browser -> Template : POST
Template -> View : post with user input
View -> Model : insert data according to user input
Model -> View : response success/error
View -> Template : display success/error
Template -> Browser : content

@enduml
```

Entwicklung der Webanwendung

in diesem abschnitt wird erläutert wie die Webanwendung zusammengesetzt ist. Wie oben bereits erwähnt kommt als Framework Django zum einsatz. Dessen aufgabe ist es die Website mit der mySQL Datenbank zu verbinden um eine Kommunikation zwischen Webanwenung und Datenbank zu ermöglichen. Die Webanwendung liest und schreibt auf der Datenbank.

Verwendete Sprachen

- HTML (Hypertext Markup Language)
- CSS (Cascading Style Sheets)
- js (Java Script)
- py (Python)

Verzeichnisstruktur

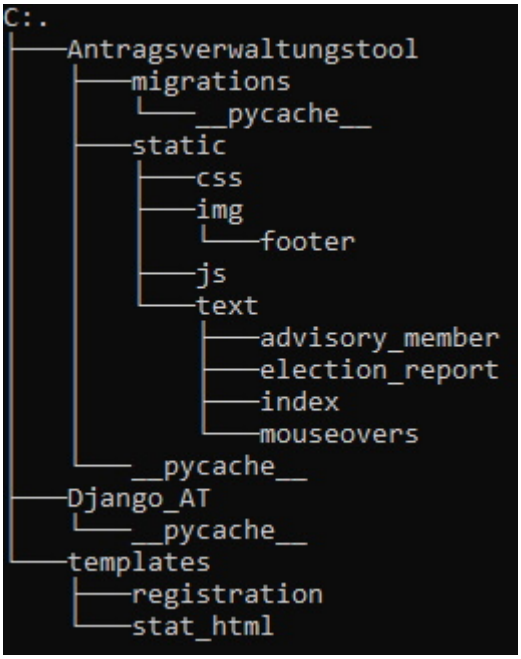


Abbildung 1. Antragsverwaltungstool (Hauptverzeichnis in der die gesammte Webanwendung und das Django Framework liegt)

HTML

Als Basis für unser System soll eine Webanwendung dienen. Diese hat den Vorteil das diese von jedem moderenen Browser der HTML 5.0 oder neuer unterstützt dargestellt werden kann. Getestet wurde sie mit den gänigsten Browsern

- Google Chrome (V. 91.0.4472.114)
- Safari (V. 14.0.3)
- Microsoft Edge (V. 91.0.864.59)
- Mozilla Firefox (V. 89.0.2)

Die Webanwendung wurde ohne einen Baukasten erstellt sondern selbst gecodet unter zuhilfenahme eines Vordefinierten Stylesheets von W3-Schools (W3.css) mit Vordefinierten Fraben, Animationen, etc.

HTML Code grundaufbau

Jedes File hat immer den gleichen aufbau:

```
<!DOCTYPE html>
<html lang="de">
    <head>
        <title>Antragsverwaltungstool</title>
        <link> Verschiedene Imports für den Style der Webanwendung
        <script></script> Verschiedene Java Srcipte für die Funktionalität der
Webanwendung
    </head>

    <body>
        <div>StuRa Logo</div>
        <div>Navigationsleiste</div>
        <div>Content</div> Hier kommt alles hin das auf der Seite angezeigt werden
soll
        <div>Footer</div>
    </body>
</html>
```

Bis auf die im Folgenden beschriebe Syntax Veränderung von Dateipfaden kann troz Framework "ganz normal" HTML gecodet werden.

Java Script und CSS sowie bilder und die Angezeigten Texte wurden in externe Dateien und Verzeichnise ausgelagert damit Sie auf jeder Seite neu als link eingebettet werden können. Damit erhält man mehr übersicht und kann einen Style der Webanwendung verwenden ohne alles auf jeder seite neu schreiben zu müssen.

Beispiel für einbinden eines CSS files das im Verzeichnis für CSS files abgelegt ist ohne Django Framework

```
<link rel="stylesheet" href="static/css/mainstyle.css">
```

Beispiel für einbinden eines CSS files das im Verzeichnis für CSS files abgelegt ist mit dem Django Framework

```
<link rel="stylesheet" href="{% static 'css/mainstyle.css' %}">
```

Durch das Framework verändert sich die Angabe von Pfaden wie im Oben gezeigten Beispiel. Das gilt auch für alle anderes Dateipfade die auf ein internes Script, Stylesheet oder eine Bildtatei verweisen sollen. Dabei ist der jeweils verwendete HTML Tag egal. Wenn man allerdings auf ein Externes Objekt verweist wie beispielweise auf eine andere Hompage dann kann der link "normal" eingefügt werden. Beispiel:

footer.html

```
<td><a href="https://www.w3schools.com/default.asp"><img src="{% static
'img/footer/W3.png' %} " alt = "W3 Logo"/><br>W3 schools</a></td>
```

Ausschnitt aus dem Footer. der link auf die Homepage von "W3 Schools" ist unverändert abgelegt aber der Dateipfad zum Logo welches auf dem Toolserver abgelegt ist ist mit der Django Syntax eingebunden worden.

Java Script

Java Script hat in dieser Webanwenung die Aufgabe Kontetnt auf wunsch des Benutzers ein- und Auszublenden das passiert mit umstylen der Aktuellen seite über CSS befehle.

myInputToggle.js

```
function togglecheck(toggleID1,radioID){
    var x = document.getElementById(toggleID1);
    if(document.getElementById(radioID).checked === true) {
        x.style.display = "none";
    }else {
        x.style.display = "block";
    }
}
```

Diese Funktion blendet ein auf knopfdruck ein und aus. Wie im Antrag auf Stelle/Amt zu sehen, "toggleID1" ist die id des zu Stylenden divs und "radioID" ist die id des zu drückenden Buttons, hier eine Checkbox, die die Bedingung für den Style wechsel ist. Wenn ".checked" erfüllt ist soll "toggleID1" ausgeblendet werden. Wenn ".checked" nicht erfüllt ist soll "toggleID1" eingeblendet werden. Das ist auch der default fall.

myInputToggle.js

```
function togglecheck_stura(toggleID1,toggleID2,radioID){
    var x = document.getElementById(toggleID1);
    var y = document.getElementById(toggleID2);
    if(document.getElementById(radioID).checked === true) {
        x.style.display = "none";
        y.style.display = "block";
    }else {
        x.style.display = "block";
        y.style.display = "none";
    }
}
```

Diese Funktion macht genau das selbe wie die Obrige Funktion "togglecheck()" nur hat diese noch dein zusatzparameter "toggleID2". Beim Ausführen der Funkton wird ein ausgeblendet (umgestylt) und ein anderes sichtbar gemacht (umgestylt)

myInputToggle.js

```
function mytoggleinputs(f) {
    var Sin = document.getElementById("Sin").value;     //Stellen Input
    var x = document.getElementById(f);
    if(Sin === "18") {
        if (x.style.visibility === "hidden") {
            x.style.visibility = "visible";
        }
    } else {
        x.style.visibility = "hidden";
    }
}
```

Auch diese Funktion arbeitet "togglecheck()" nur das hier nicht ewtwas ne displayt wird sondern die Sichtbarkeit verändert wird. Das passiert wenn man bei den Anträgen "Zuständige Stelle wählen:" auf den Reiter "Eigene" drückt. Das Dropdownmenu hat die id "Sin" und "f" ist das Objekt dessen sichtbarkeit Manipuliert wird. in "Sin" steht das Value welches der Benutzer ausgewählt hat wenn dieses Value gleich 18 (das ist das Value für den "Eigene" reiter) ist dann wird das zu Manipulierende Objekt sichtbar gestylt und wenn es ungleich 18 ist dann bleibt es weiterhin unsichtbar.

toggledisplay.js

```
function toggledisplay(f){
    var x = document.getElementById(f);
    var Sin = document.getElementById("Sin").value;     //Stellen Input
    if(Sin === "18") {
        if (x.style.display === "none") {
            x.style.display = "block";
        }
    }else {
        x.style.display = "none";
    }
}
```

Diese Funktion ist eine Mixtur aus "togglecheck()" und "mytoggleinputs()". Angewendet wird sie auf der Seite "Intern" und bledet das feld "Eigene" aus und ein. Display: none ist der Default fall. Die Mixtur besteht aus der Logik von "togglecheck()" und dem Anwendungsbereich von "mytoggleinputs()"

CSS

Die generelle Logik der Sheetaufteilung

- mainstyle
    - Alle generellen styl entscheidungen wie Hindergrundfarbe, Farbe der Navigationsleiste, Footer, etc.

- W3
  - Vordefiniertes Stylsheet von "W3 Schools" mit vielen Funktionen, Farben, Animationen, etc.
- popup
  - Style für die kleinen Mouseover "?" in den Anträgen
- formular
  - Der gennerelle Style der Anträge
- login
  - Der Stlye der Loginpage mit einem Angepassten Footer
- index, intern
  - Spezifische Styls für die jeweiligen Seiten "Index(Home)" und "Intern"

Im weiteren werden nur die wichtigsten styleelemente genauer beschrieben da es sich um gewöhnliches css styling handelt

mainstyle.css

```
body{
    background: gainsboro;
    font-family: Verdana, sans-serif;
}

.sturalogo{
    margin-top: 15px;
    margin-left: 15px;
    margin-bottom: 5px;
}
```

Style des <body> tags mit einer Hintergrundfarbe und einer Schriftart die nach dem include des css files in das HTML Dokument einen einheitlichen style der verschiedenen Seiten gewährleistet. Genau wie der <body> tag wurde eine Klasse mit dem Namen "sturalogo" erstellt und an ein gebunden mit dies stellt den Style des Tool logos das das auf jeder Seite der Anwendung links oben gefunden werden kann. Das "margin" Atribut beschreibt den Abstand des Objektes von einem Anderen Objekt bzw. Abstand zum Bildschirm-/Fensterrand.

TIP	Beim stylen von CSS kann man einem <tag> eine Klasse <tag class="myclass"> eine ID <tag id="myid"> geben. Es ist ebenfalls möglich einem <tag> beides zu geben <tag class="myclass" id="myid"> sowie mehrere Klassen und IDs <tag class="myclass1 myclass2" id="myid1 myid2">

WARNING	Wenn man die selbe id in dem selben dokument mehrfach verwendet wie beispielsweise das in der Navigationsleiste mit der "id="active" gemacht wurde dann gilt es zu beachten das diese id **NICHT** in einem Java Script verwerwendet wird. Das Problem hierbei ist das dann das Script keinen Effekt hat. Der Browser gibt auch einen Error aus das eine ID doppelt verwendet wurde in einem Dokument.

mainstyle.css

```
.mainnav{                               /*Style der Navigationsleiste*/
    list-style-type: none;
    margin-top: 5px;
    padding: 0;
    overflow: hidden;
    background-color: #333;
}

.lil{                                   /*Linkes Listenobjekt der Navigationsleite*/
    float: left;
}

.lir{                                   /*Rechtes Listenobjekt der Navigationsleiste*/
    float: right;
}

#active{                                /*Cursor der Navigationsleiste für die
aktuelle Seite*/
    background-color: #F2AC2E;
}

.mainnav a{                             /*Style des Link tags innerhalb der
Navigationsklasse*/
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

.lil a:hover:not(.active){              /*Style des Link tags innerhalb der
Navigationsklasse*/
    background-color: #F2AC2E;
}

.lir a:hover:not(.active){
    background-color: cornflowerblue;
}

.dropdown:hover .dropdown-content {     /*Style des Dropdowns der Navigationsleiste*/
    display: block;
```

```
}

.dropdown-content {                    /*Style des Dropdowns der Navigationsleiste*/
    display: none;
    position: absolute;
    background-color: #333;
    min-width: 160px;
    z-index: 1;
}
```

TIP

Das Stylen von <tags> in CSS bewerkstelligt man in dem man den Namen des <tags> in das Stylesheet oder den <style></style> tag direkt im HTML File scheibt. Beispiel: der oben gezeigte body tag. eine Klasse wird mit einem .Klassennamen und eine ID mit #IDname gestylt. Beislpiele hier: .mainnav oder #active. Ebenso ist es möglich wenn eine Klasse oder ID einen Bereich umspannt beispielsweise mit einem tag das man innerhalb dieser Klasse oder ID einen tag stylt. Damit kann man Klassen und ID Namen sparen, bzw. erhält mehr übersicht. Beispiel hier: .mainnav(Klassenname) a(der link tag). Damit stylt man den <a> tag nur innerhalb der Klasse mainnav.

WARNING

Das File W3.css ist von eine Library der Seite W3 Schools (https://www.w3schools.com/default.asp) welche ich auf grund der Verfügbarkeit Kopiert habe. Für nähere Informationen und Anwendung der verschieden Klassen, Animationen, Farben, etc. Kann unter https://www.w3schools.com/w3css/defaulT.asp genauer nachgelesen werden. Dort findet man Erklärungen und Anwendungs Tutorials.

Texte

Die verschiedenen Texte der Webanwendung wie beispielsweise auf der Homeseite oder die Texte der Mouseovers in den Anträgen sowie die verschieden Fragen wurden nicht in die Verschieden Files geschrieben sondern ausgelagert. Damit bietet sich der Vorteil das die Texte schnell und einfach Angepasst werden können. Die Technik dahinter ist eine Java Script datei die einen String beinhaltet welcher dann ausgegeben wird.

email.js

```
var text = 'Bei möglichkeit die HTW-Mail verwenden (sXXXXX@htw-dresden.de)';

    document.writeln(text);
```

TIP

Wenn der Text zu lang ist das er den Rahmen einer Zeile sprent kann der Text wie in Java gesplittet werden. Beispiel wäre der Text aus dem financestext.js im text Verzeichnis unter index

TIP	In den 'String' des Text-Java Scrips kann Problemlos auch HTML code untergebracht werden. Beispiel wäre der Text aus dem welcometext.js im text Verzeichnis unter index

Textordner struktur und Namenslegende

- advisory_member = Beratendes Mitglied
  - frage 1, 2, 3, 4 = Die vier Fragen im Formular die Kursiv gedruckt sind
- election_report = Stelle/Amt
  - die ersten vier Fragen aus "advisory_member" kommen zu erst danach
  - frage 1, 2, 3 = Diese bilden dann Frage 5-7
- index = Indexseite/Home
  - advisorytext = Beratendes Mitglied Box
  - electiontext = Stelle/Amt Box
  - establishing = Herstellung des Benehmen Box
  - financestext = Antrag mit finanziellen Mitteln Box
  - universaltext = Antrag ohne finanziellen Mitteln Box
  - welcometext = StuRa Antragsverwaltung Box
- mouseovers
  - antragtext = Fragezeichen Mausover an jeder Box "Antragstext"
  - email = Fragezeichen Mausover an jeder Box "E-Mail"
  - KostenpositionimHaushaltsplantext = Fragezeichen Mausover an der Box "Kostenposition im Haushaltspan" im "Antrag mit finanziellen Mitteln"