

HANDBUCH ZUM NUTZUNG SERVER-KOMPONENTE UND CLIENT-KOMPONENTE

Inhaltsverzeichnis:

- Handbuch zum Nutzung der Website "PARLIAMENT SENTIMENT RADAR"
 - Einleitung
 - Borwort
 - Vorbereitung
 - Frontend
 - Chart Visualisation
 - Filter Funktion
 - Volltext Funktion
 - Backend
 - XML File einzulesen
 - Methode von MongoDB
 - Erstellen
 - Einschreiben
 - Lesen
 - Updaten
 - Löschen
 - RESTful Schnittstelle
 - Return protocols
 - Return speakers
 - Return speeches
 - Return fractions
 - Return annotations

EINLEITUNG

Vorwort

Die Anwendung ist browserbasiert. Die Administration läuft in folgenden Browsern:

- 1 - Internet Explorer
- 2 - Mozilla Firefox
- 3 - Safari
- 4 - Chrome

Dieses Handbuch ist zur Erläuterung der Verwendung des gemeinschaftliche Gruppenabschlussprojekt PRG-Praktikum WiSe 2021/2022 "Parliament Sentiment Radar" zuständig.

Die Lösung dieses Gruppenprojekts haben sich die Teilnehmerinnen jeweils in Gruppen zusammengefunden. Hierzu verwenden wir, wie auch schon in den vorherigen Übungen, GitLab.

Um alle Anwendungsfunktionen nutzen zu können, müssen Sie in Ihren Browsereinstellungen Folgendes zulassen:

- 1 - Ausführung von Java Script
- 2 - Verwendung von Sitzungscookies
- 3 - Öffnen von Popup-Fenstern

Vorbereitung

▼ Hide Information

Unsere Gruppe hat zur Bearbeitung der Abschlussaufgabe ein privates Projekt namens "Gruppe_8_Mittwoch_3_ParliamentSentimentRadar" in Gitlab gestellt und hat unserer Tutorin das Projekt mit der Berechtigung lesend frei gegeben.

Alles JavaScript-Dateien wurden ausführlich dokumentiert.

Das "Parliament Sentiment Radar" besteht aus zwei Komponenten: einer Server-Komponente, implementiert in Java, sowie einer Client-Komponente, implementiert in JavaScript unter Verwendung von SB Admin 2 und beinhaltet folgenden Funktionalitäten:

- 1 1. Einlesen von Parlamentsdebatten des Deutschen Bundestages im XML-Format.
- 2 2. Analysieren, Aufbereiten und datenbankgestütztes Abspeichern der abgefragten Parlamentsreden.
- 3 3. Verarbeitung der Parlamentsreden mittels angegebener NLP-Verfahren.
- 4 4. Datenbankgestütztes Abfragen und Auswerten der Ergebnisse.
- 5 5. Implementierung eines RESTful Webservice zur Bereitstellung der Parlamentsreden für die Client-Komponente.
- 6 6. Implementierung einer Visualisierungskomponente unter Verwendung von SB Admin 2 gemäß Aufgabe 3.

Unsere Server-Komponente ist unter Verwendung von Java Spark als RESTful Webservice implementiert.

Zum Testen und Dokumentieren der REST-Abfragen verwenden wir **swagger.io**.

FRONTEND

Die Hauptaufgabe von Frontend-Teil dient dazu, die Ergebnisse mittels SB Admin 2 gemäß der definierten Aufgaben zu visualisieren und analysieren. Es gibt zwei Websites in unsere Frontend-Teil.

Main Website - Dashboard

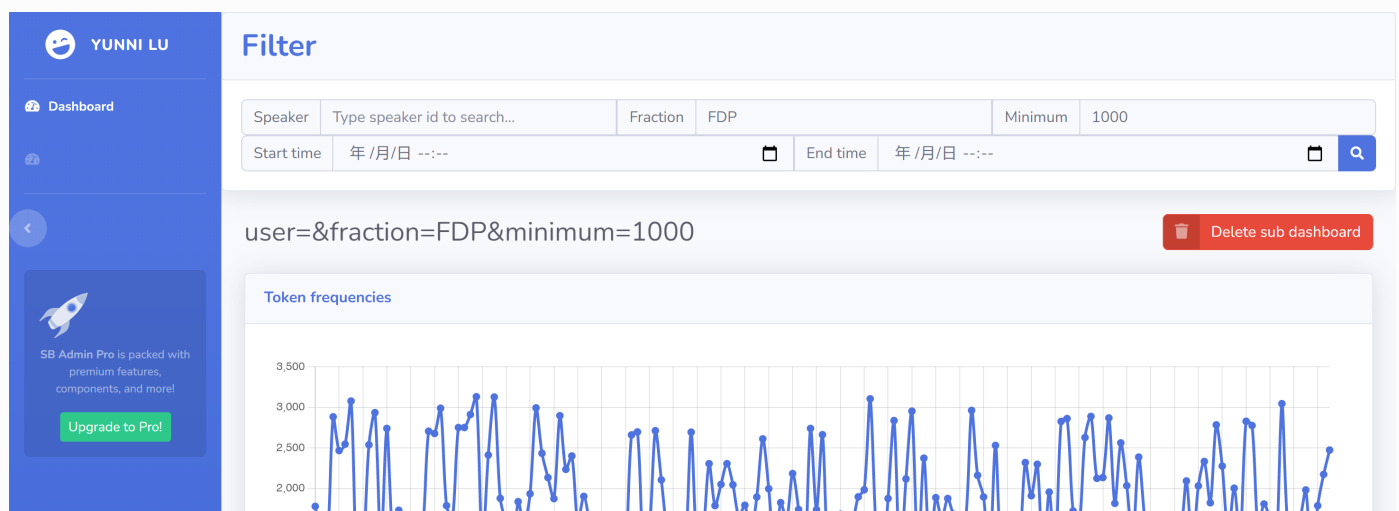


Chart Visualisation

Das Dashboard "**Übersicht**" visualisiert die gesamte Parlament Daten aus unsere Datenbank:

1	- Tokenchart
2	- Poschart
3	- Sentimentchart
4	- Named Entities chart
5	- Rednerchart mit Redner-Bild als Hover-Effekt

Filter Funktion

Man kann im oberen Seitenbereich je nach dem Suchfeld frei entscheiden, ob man durch Dropdown-Menü oder selbst tippen, die entsprechenden Filterergebnisse suchen:

- Speaker-ID (Format: Dropdown-Menü oder 8-stellige ID-Zahlen wie z.B. 11004908)

Speaker	Type speaker id to search...
---------	------------------------------

- Fraktionsnamen, hier ist in dem Bild als Beispiel "FDP" geschrieben (Format: Dropdown-Menü oder Fraktionszugehörigkeit)

Fraction	FDP
----------	-----

- minimale Anzahl z.B. für Token, Pos usw. hier ist in dem Bild als Beispiel "1000" geschrieben (Format: Dropdown-Menü oder 3 bis 4-stellige Zahlen)

Minimum	1000
---------	------

- Beginn- und End-Zeitraum (Format: Dropdown-Menü)

Start time	年 /月/日 --:--	📅	End time	年 /月/日 --:--	📅
------------	--------------	---	----------	--------------	---

Es wird nach Klicken



ein sogenannte "**Sub-Dashboard**" angezeigt (mit

Suchanfragetitel), dies enthält die gleichen Visualisierungen und Optionen wie dem "Main" Dashboard, jedoch nur für die Suchanfrage.

In dem Bild sieht man, dass man kein SpeakerID aber Fraktion "FDP" und einen minimale Anzahl "1000" eingegeben hat.

```
user=&fraction=FDP&minimum=1000
```

Das Sub-Dashboard kann durch Button "**Delete sub dashboard**" wieder entfernt werden.

Sub Website - Reden

Volltext Funktion

Links bei der Sidebar unter "Dashboard" sieht man die Website "reden". Die Volltext-Visualisierung ist für jede einzelne Parlaments-Rede implementiert und beinhaltet folgenden Funktionen:

GRUPPE
8_MITTWOCHE_32

Dashboard

Reden

Reden-Visualisierung

Filter mit Tagesordnungspunkten

Wahlperiode 20
Sitzung 11
Tagesordnungspunkte Tagesordnungspunkt 7
Rede

ID20206200

Name: Marc Bernhard, Fraktion: AfD

Person
Ort
Organisation

Frau Präsidentin! Meine Damen und Herren! **Angela Merkel** hat in 16 Jahren dafür gesorgt, dass wir in **Deutschland** die Strompreise der Welt haben, doppelt so hoch wie in unseren Nachbarländern. Und Sie von der vielleicht künftigen Bundesregierung wollen das jetzt sogar noch toppen und uns Bürger mit noch höheren Energiepreisen abzocken. Frau Präsidentin! Meine Herren! **Angela Merkel** hat in 16 Jahren dafür gesorgt, dass wir in **Deutschland** die höchsten Strompreise der Welt haben, so hoch wie in unseren Nachbarländern. Und Sie von der vielleicht künftigen Bundesregierung wollen das jetzt sogar noch toppen und uns Bürger mit noch höheren Energiepreisen abzocken. So kostet heute schon Heizöl mehr als das Doppelte als im letzten Jahr und die Explosion der Gaspreise macht den Menschen im **Land Angst**. Eine vierköpfige Familie zahlt über 600 Euro im Monat für Strom, Benzin und Heizung. (Dr. Götz Frömming [AfD]: Wahnsinn!) also 1 800 Euro pro Jahr mehr als letztes Jahr. Dafür verantwortlich sind Bundesregierungen, die in maßloser Selbstüberschätzung im nationalen Alleingang angeblich die Welt retten wollen. (Beifall bei der AfD – Kai Whittaker [CDU/CSU]: Sie finden doch nationale Alleingänge immer super!) Sie sorgen dafür, dass das Heizen in diesem Winter für viele Menschen unbezahlbar wird. Bereits im letzten Winter mussten mehr als 7 Millionen Menschen frieren, weil sie ihre Wohnung nicht mehr ausreichend heizen konnten. Diese soziale Kälte der Regierung trifft besonders Bedürftige, Rentner, Geringverdienende und Alleinerziehende. Und SPD-Spitzengehirne wie **Katarina Barley** fällt nichts ein, als die Menschen zu verhöhnen und ihnen zu raten, dass sie doch einfach weniger Strom verbrauchen und weniger heizen sollen, wenn sie sich die Energiepreise nicht mehr leisten können. (Zuruf des Abg. **Kevin Kühnert** [SPD]) Dabei sind die hohen Energiepreise doch gerade das direkte Ergebnis Ihrer völlig verfehlten Klimapolitik, die jetzt von der Realität eingelenkt und nichts anderes. (Beifall bei der AfD) Mit der Energiesteuer, der Ökosteuer, der EEG-Umlage, der **Gassteuer**, der **Mineralölsteuer**, der Stromsteuer, der CO2-Steuer usw. pressen Sie uns Bürgern immer neue und höhere Belastungen auf. Während Sie unsere Bürger immer weiter abzocken und in kalten Wohnungen sitzen lassen, senkt **Italien** die Energiepreise für seine Bürger – und zwar mit deutschen Steuergeldern aus den EU-Coronahilfen, meiner sehr geehrten Damen und Herren bei der AfD – Zurufe von der SPD) Es ist wirklich eine Schande, dass die Bundesregierung Hunderte von Milliarden Euro ins Ausland verschenkt, während Millionen von Menschen in unserem Land frieren müssen. Damit muss endlich Schluss sein. (Beifall bei der AfD) Deshalb fordern wir in unserem Antrag auch die sofortige Streichung der EEG-Umlage, die Abschaffung der Energiesteuer und die sofortige Aussetzung der Mehrwertsteuer auf Strom und Heizkosten. Herzlichen Dank, meine sehr geehrten Damen und Herren. (Beifall bei der AfD – **Christian Kühn** [Tübingen] [BÜNDNIS 90/DIE GRÜNEN]: So ein Bullshit! – Gegenruf von **Marc Bernhard** [AfD]: Was stimmt daran nicht? Müssen 10 Millionen Menschen frieren, oder nicht?) Vizepräsidentin **Yvonne Magwas**: Ihre Redezeit ist vorbei, Herr **Bernhard**. – Ich erteile das Wort zu seiner ersten Rede im **Deutschen Bundestag** **Kühnert**.

Alle im Text gefundenen Named Entities werden farblich hinterlegt.

Person

Ort

Organisation

- Der Sprech*innenname, Informationen zu seiner/ihrer Partei- und Fraktionszugehörigkeit, sowie ein Foto(sofern vorhanden) werden im Kopf der Rede angezeigt. (Bei dem obigen Beispiel ist der Abgeordnete Name: Marc Bernhard, Fraktionszugehörigkeit: AfD)
- Ermöglichung eines Filters über eine Navigation durch die Tagesordnung der Protokolle.

-

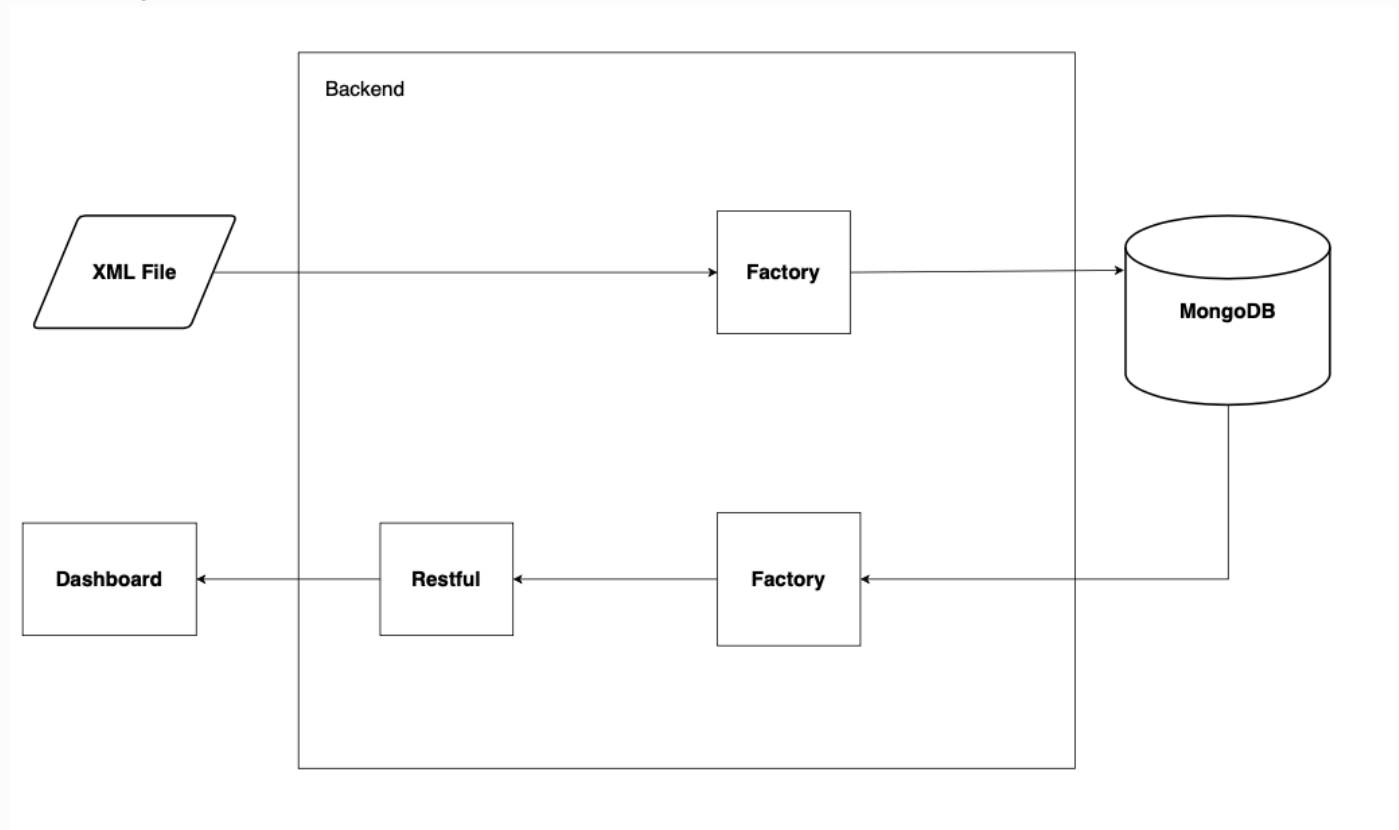
Filter mit Tagesordnungspunkten

Wahlperiode	20 ▼	Sitzung	11 ▼	Tagesordnungspunkte	Tagesordnungspunkt 7 ▼	Rede
ID20206200 ▼	🔍					

- Man wählt mit dem DropDown-Menü in der Reihenfolge:
 1. Wahlperiode
 2. Sitzung
 3. Tagesordnungspunkte
 4. RedeID

BACKEND

Flussdiagramm für Backend:



Zuerst lesen die XML File ein. Dann verwendet Factory, um die Daten in unsere eigene Datenstruktur zu speichern. Dann schreiben die Daten in MongoDB ein. Verwendet wieder die Fabrik, um Daten aus MongoDB zu lesen. Und implementiert Restful -Schnittstelle. Auf diese Weise kann der Frontend über diese Restful-Schnittstelle visualisiert werden.

Funktionsweise von Backend ist nicht relevant für Benutzer. Deshalb werden hier keine Details genannt. Die Details werden während des Vorrechnen erläutert.

XML File einzulesen

Führt die *Main-Methode* der ProtocolMongoDBWriter-Klasse aus. Der Object *handler* für die Klasse MongoDBConnectionHandler wird über die Main-Methode erstellt. Wir können dann eine Verbindung zu MongoDB herstellen. Die Parlamentstexte werden direkt vom Bundestag heruntergeladen. Die XML-Datei wird dann über ParliamentFactory eingelesen.

```

1 public static void main(String[] args) {
2     MongoDBConnectionHandler handler = new
MongoDBConnectionHandler("config/config.json");
3     String protocolDirectory = "Daten/20. Wahlperiode";
4     ParliamentFactory factory = new ParliamentFactory_Impl();
5     factory.initFromDirectory(protocolDirectory);
6     //     writeProtocols(factory.getProtocols(), handler);
7     writeSpeakers(factory.getParliamentMembers(), handler);
8     //     writeSpeechs(factory.getSpeeches(), handler);
9     //     writeFractions(factory.getFractions(), handler);
1    }

```

Methode von MongoDB

Über das Objekt handler von MongoDBConnectionHandler kann man folgende Methode verwenden.

Diese Methode werden in Klasse MongoDBConnectionHandler gespeichert.

- Erstellen
- Einschreiben
- Lesen
- Updaten
- Löschen

Erstellen

Wenn wir das Project ausführen, wird MongoDBConnectionHandler schon erstellt.

Einschreiben

1.Schreib nur ein Dokument in MongoDB ein:

Verwenden diese Methode *writeDocument* von Klass MongoDBConnectionHandler:

```

1 public void writeDocument(String collectionName, Document
  document){
2     try{
3         MongoCollection<Document> collection =
  this.getCollection(collectionName);
4         collection.insertOne(document);
5     } catch (Exception e){
6         e.printStackTrace();
7     }
8 }

```

Mit diesem Befehl kann man einen Dokument in eine spezielle Collection in MongoDB einschreiben.

```

1 handler.writeDocument(collectionName, document);

```

2.Schreib mehrerer als eins Dokumente in MongoDB ein:

Verwenden diese Methode *writeDocuments* von Klass

MongoDBConnectionHandler:

```

1 public void writeDocuments(String collectionName,
  List<Document> documents){
2     try{
3         MongoCollection<Document> collection =
  this.getCollection(collectionName);
4         collection.insertMany(documents);
5     } catch (Exception e){
6         e.printStackTrace();
7     }
8 }

```

Mit diesem Befehl kann man mehrere Dokumente in eine spezielle Collection in MongoDB einschreiben.

```

1 handler.writeDocuments(collectionName, documents);

```

Lesen

Suche nach einem bestimmten Dokument in einer bestimmten Collection in MongoDB.

Verwenden diese Methode *getDocument* von Klasse MongoDBConnectionHandler:

```
1 public Document getDocument(String collectionName, Bson
  query){
2     try{
3         MongoCollection<Document> collection =
  this.getCollection(collectionName);
4         return collection.find(query).first();
5     } catch (NullPointerException e){
6
7         System.err.println("MongoDBConnectionHandler.getDocument:
  collection " + collectionName + " not found!");
8         return null;
9     } catch (Exception e) {
10        e.printStackTrace();
11        return null;
12    }
13 }
```

Kann man durch diesem Befehl erreichen:

```
1 handler.getDocument(collectionName, query);
```

Updaten

Mit diesem Method kann man die Dokument, der in eine spezielle Collection in MongoDB ist, updaten.

Verwenden diese Methode *updateDocument* von Klasse MongoDBConnectionHandler:

```
1 public boolean updateDocument(String collectionName, Bson
  query, Document newDocument){
2     MongoCollection<Document> collection =
  this.getCollection(collectionName);
3     try{
```

```

4      UpdateResult result = collection.replaceOne(query,
newDocument);
5      if (result != null && result.getModifiedCount() == 0 &&
result.getMatchedCount() == 0){
6          try {
7
8              this.getCollection(collectionName).insertOne(newDocument);
9              return true;
10             }
11             catch (Exception e){
12                 System.out.println(e.getMessage());
13             }
14         }
15     } catch (Exception e){
16         e.printStackTrace();
17     }
18     return false;
19 }

```

Kann man mit diesem Befehl erreichen:

```

1 handler.updateDocument(collectionName, query, newDocument);

```

Wenn die Dokument erfolgreich updatet wird, geben true aus.

Löschen

Mit diesem Method kann man die Dokument, der in eine spezielle Collection in MongoDB ist, löschen.

Verwenden diese Methode *deleteDocument* von Klass

MongoDBConnectionHandler:

```

1 public Boolean deleteDocument(String collectionName, Bson
  query){
2     try {
3         MongoCollection<Document> collection =
  this.getCollection(collectionName);
4         DeleteResult result = collection.deleteOne(query);
5         System.out.println("Deleted document count: " +
  result.getDeletedCount());
6         return true;
7     } catch (Exception e) {
8         // System.err.println("Unable to delete due to an error:
  " + e);
9         e.printStackTrace();
1        return false;
10    }
11 }

```

Kann man mit diesem Befehl erreichen:

```

1 handler.deleteDocument(collectionName, query);

```

Wenn die Dokument erfolgreich gelöscht wird, geben true aus.

RESTful Schnittstelle

Wenn die XML File eingelesen und NLP analysiert werden, werden durch ParliamentFactory in MongoDB eingeschrieben. Aber das Frontend kann jedoch keine Daten direkt von MongoDB für die Visualisierung abrufen. Deshalb verwenden wir hier RESTful Schnittstelle.

Hier ist unsere Swagger.

Swagger Parliament API1.0.0

[Base URL: 38.242.210.53:4567]

./swagger.yml

Parliament API – Gruppe_8_mittwoch_3

Contact the developer

Apache 2.0

Find out more about Swagger

Schemes

HTTP

protocolReturn protocols

GET /protocol

GET /protocols

GET /agendaitem

GET /agendaitems

speakerReturn speakers

GET /speakers

speechReturn speeches

GET /speeches

GET /speech

fractionReturn fractions

GET /fractions

GET /fraction

annotationReturn annotations

GET /sentiment

GET /tokens

GET /pos

GET /namedEntities

GET /dependencies

statisticReturn statistics for speakers and speeches

GET /statistic

Return protocols

Get protocol:

Sollt man hier zwei Parameters *session* und *term* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

The interface shows a GET request to the endpoint `/protocol`. Below the endpoint, there is a 'Parameters' section with a 'Try it out' button. The parameters are:

Name	Description
session ★ required string (query)	session of the protocol
term ★ required string (query)	term of the protocol

Input fields for 'session' and 'term' are provided, with the values 'session' and 'term' entered respectively.

Struktur der Ergebnisse:

The 'Responses' section shows a 200 status code for a 'successful operation'. The 'Response content type' is set to 'application/json'. The 'Example Value' is displayed in a dark box with the following JSON structure:

```
{
  "date": "24.10.2017",
  "startTime": "2017-10-24T11:00:00.000+02:00",
  "endTime": "2017-10-24T17:03:00.000+02:00",
  "session": "1",
  "term": "19",
  "agendaItemIds": [
    "Tagesordnungspunkt 1",
    "Tagesordnungspunkt 2",
    "Tagesordnungspunkt 3",
    "Tagesordnungspunkt 4",
    "Tagesordnungspunkt 5",
    "Tagesordnungspunkt 6"
  ]
}
```

Get protocols:

Hier müssen keine Parameter eingegeben werden.

Klicken Sie auf die Schaltfläche *Try it out*:

The interface shows a GET request to the endpoint `/protocols`. Below the endpoint, there is a 'Parameters' section with a 'Try it out' button. The parameters section is empty, indicating 'No parameters'. The 'Responses' section at the bottom shows a 'Response content type' of 'application/json'.

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
Example Value Model	
<pre>{ "result": [{ "date": "24.10.2017", "startTime": "2017-10-24T11:00:00.000+02:00", "endTime": "2017-10-24T17:03:00.000+02:00", "session": "1", "term": "19", "agendaItemIds": ["Tagesordnungspunkt 1", "Tagesordnungspunkt 2", "Tagesordnungspunkt 3", "Tagesordnungspunkt 4", "Tagesordnungspunkt 5", "Tagesordnungspunkt 6"] }], "success": true}</pre>	

Get agendaitem:

Sollt man hier drei Parameters *session*, *term* und *id* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET /agendaitem

Try it out

Parameters

Name	Description
session <small>★ required</small> string (query)	session of the protocol
<input type="text" value="session"/>	
term <small>★ required</small> string (query)	term of the protocol
<input type="text" value="term"/>	
id <small>★ required</small> string (query)	id of the agendaitem
<input type="text" value="id"/>	

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
Example Value Model	
<pre>{ "id": "Tagesordnungspunkt 1", "protocolId": [1, 19], "speechIds": ["ID1910000100"]}</pre>	

Get agendaitems:

Sollt man hier zwei Parameters *session* und *term* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/agendaitems

Parameters

Try it out

Name	Description
session <small>★ required</small> string <small>(query)</small>	session of the protocol
<input type="text" value="session"/>	
term <small>★ required</small> string <small>(query)</small>	term of the protocol
<input type="text" value="term"/>	

Struktur der Ergebnisse:

Responses

Response content type application/json

Code	Description
200	successful operation
Example Value Model	
<pre>{ "result": [{ "id": "Tagesordnungspunkt 1", "protocolId": [1, 19], "speechIds": ["ID1910000100"] }], "success": true }</pre>	

Return speakers

Get speakers:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.
Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/speakers

Parameters

Try it out

Name	Description
<div>user</div> <div>string</div> <div>(query)</div>	id of the speaker
<div>fraction</div> <div>string</div> <div>(query)</div>	name of the fraction
<div>minimum</div> <div>integer</div> <div>(query)</div>	minimum frequency
<div>time[gte]</div> <div>string</div> <div>(query)</div>	time range to filter speeches
<div>time[lte]</div> <div>string</div> <div>(query)</div>	time range to filter speeches

Struktur der Ergebnisse:

Responses

Response content type

application/json

Code	Description
200	successful operation

Example Value

Model

```
{
  "result": [
    {
      "firstname": "Diether",
      "name": "Dehm",
      "id": "11000365",
      "titel": "Dr.",
      "fraction": "DIE LINKE",
      "role": null,
      "image": {
        "url": "https://bilddatenbank.bundestag.de/fotos/file7e1xgdko1bk63jb111l.jpg",
        "description": "Dehm, Diether Beschreibung: Dr. Diether Dehm, DIE LINKE., Bundestagsabgeordneter, Abgeordneter, Mitglied des Deutschen Bundestages, MdB, Rede, Deutscher Bundestag, 132. Sitzung, TOP 1, Thema: Europ ischer Rat und EU-Reformvertrag, Redner 10.; Rednerpult.\n\rFotograf/in: Lichtblick/Achim Melde"
      },
      "speechIds": [
        "ID1910206100"
      ]
    }
  ],
  "success": true
}
```

Return speeches

Get speeches:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.
Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/speeches

^

Parameters

Try it out

Name	Description
user string (query)	id of the speaker
	<input type="text" value="user"/>
fraction string (query)	name of the fraction
	<input type="text" value="fraction"/>
minimum integer (query)	minimum frequency
	<input type="text" value="minimum"/>
time[gte] string (query)	time range to filter speeches
	<input type="text" value="time[gte]"/>
time[lte] string (query)	time range to filter speeches
	<input type="text" value="time[lte]"/>

Struktur der Ergebnisse:

Responses

Response content type application/json

Code	Description
200	successful operation, return list of speech ids
	Example Value Model
	<pre>{ "result": ["string"], "success": true }</pre>

Get speech:

Sollt man hier ein Parameter *id* eingeben.
Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/speech

^

Parameters

Try it out

Name	Description
id * required string (query)	The name that needs to be fetched. Use user1 for testing.
	<input type="text" value="id"/>

Struktur der Ergebnisse:

Responses

Response content type application/json

Code	Description
200	successful operation

Example Value | Model

```
{  "date": "15.05.2019",  "protocolId": [    [      1,      19    ]  ],  "speaker": "11004323",  "id": "ID1910000100",  "texts": [    {      "id": "Sehr geehrter Herr Pr?sident! Liebe Kolleginnen und Kollegen!",      "name": "J_1"    }  ],  "annotations": {    "persons": [      "Bundesminister Karliczek"    ],    "locations": [      "Deutschland"    ],    "organisations": [      "Europ?ische Union"    ]  }
```

Return fractions

Get fractions:

Hier m?ssen keine Parameter eingegeben werden.

Klicken Sie auf die Schaltfl?che *Try it out*:

GET /fractions

Parameters

Try it out

No parameters

Struktur der Ergebnisse:

Responses

Response content type application/json

Code	Description
200	successful operation

Example Value | Model

```
{  "result": [    {      "id": "string",      "members": 0,      "memberIds": [        "string"      ]    }  ],  "success": true}
```

Get fraction:

Sollt man hier ein Parameter *name* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/fraction

Parameters

Try it out

Name	Description
name <small>★ required</small>	Fraction name
string (query)	<input type="text" value="name"/>

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
	Example Value Model
	<pre>{ "id": "string", "members": 0, "memberIds": ["string"] }</pre>

Return annotations

Get sentiment:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.
Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/sentiment

Parameters

Try it out

Name	Description
user string (query)	id of the speaker
	<input type="text" value="user"/>
fraction string (query)	name of the fraction
	<input type="text" value="fraction"/>
minimum integer (query)	minimum frequency
	<input type="text" value="minimum"/>
time[gte] string (query)	time range to filter speeches
	<input type="text" value="time[gte]"/>
time[lte] string (query)	time range to filter speeches
	<input type="text" value="time[lte]"/>

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
	Example Value Model
	<pre>{ "result": [{ "count": 0, "sentiment": "string" }], "success": true }</pre>

Get tokens:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.
Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/tokens

^

Parameters

Try it out

Name	Description
user string (query)	id of the speaker
	<input type="text" value="user"/>
fraction string (query)	name of the fraction
	<input type="text" value="fraction"/>
minimum integer (query)	minimum frequency
	<input type="text" value="minimum"/>
time[gte] string (query)	time range to filter speeches
	<input type="text" value="time[gte]"/>
time[lte] string (query)	time range to filter speeches
	<input type="text" value="time[lte]"/>

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
	Example Value Model
	<pre>{ "result": [{ "token": "string", "count": 0 }], "success": true }</pre>

Get pos:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.
Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/pos

Parameters

Try it out

Name	Description
user string (query)	id of the speaker
	<input type="text" value="user"/>
fraction string (query)	name of the fraction
	<input type="text" value="fraction"/>
minimum integer (query)	minimum frequency
	<input type="text" value="minimum"/>
time[gte] string (query)	time range to filter speeches
	<input type="text" value="time[gte]"/>
time[lte] string (query)	time range to filter speeches
	<input type="text" value="time[lte]"/>

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
	Example Value Model
	<pre>{ "result": [{ "pos": "string", "count": 0 }], "success": true }</pre>

Get namedEntities:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.
Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET

/namedEntities

^

Parameters

Try it out

Name	Description
user string (query)	id of the speaker
	<input type="text" value="user"/>
fraction string (query)	name of the fraction
	<input type="text" value="fraction"/>
minimum integer (query)	minimum frequency
	<input type="text" value="minimum"/>
time[gte] string (query)	time range to filter speeches
	<input type="text" value="time[gte]"/>
time[lte] string (query)	time range to filter speeches
	<input type="text" value="time[lte]"/>

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
	Example Value Model
	<pre>{ "result": { "persons": [{ "element": "string", "count": 0 }], "organisations": [{ "element": "string", "count": 0 }], "locations": [{ "element": "string", "count": 0 }] } }</pre>

Get dependencies:

Sollt man hier fünf Parameters *user*, *fraction*, *minimum*, *time* and *time* eingeben.

Nach Eingabe der Parameter klicken Sie auf die Schaltfläche *Try it out*:

GET /dependencies

Parameters

Try it out

Name	Description
user string (query)	id of the speaker
	<input type="text" value="user"/>
fraction string (query)	name of the fraction
	<input type="text" value="fraction"/>
minimum integer (query)	minimum frequency
	<input type="text" value="minimum"/>
time[gte] string (query)	time range to filter speeches
	<input type="text" value="time[gte]"/>
time[lte] string (query)	time range to filter speeches
	<input type="text" value="time[lte]"/>

Struktur der Ergebnisse:

Responses

Response content type application/json

Code	Description
200	successful operation
	Example Value Model
	<pre>{ "result": [{ "dependency": "string", "count": 0 }], "success": true }</pre>

Return statistics for speakers and speeches

Get statistic:

Hier müssen keine Parameter eingegeben werden.

Klicken Sie auf die Schaltfläche *Try it out*:

GET /statistic

Parameters

Try it out

No parameters

Struktur der Ergebnisse:

Responses

Response content typeapplication/json

Code	Description
200	successful operation
	Example Value Model
	<pre>{ "result": { "speakers": [{ "count": 0, "id": "string" }], "speeches": [{ "length": 0, "id": "string" }] }, "success": true}</pre>