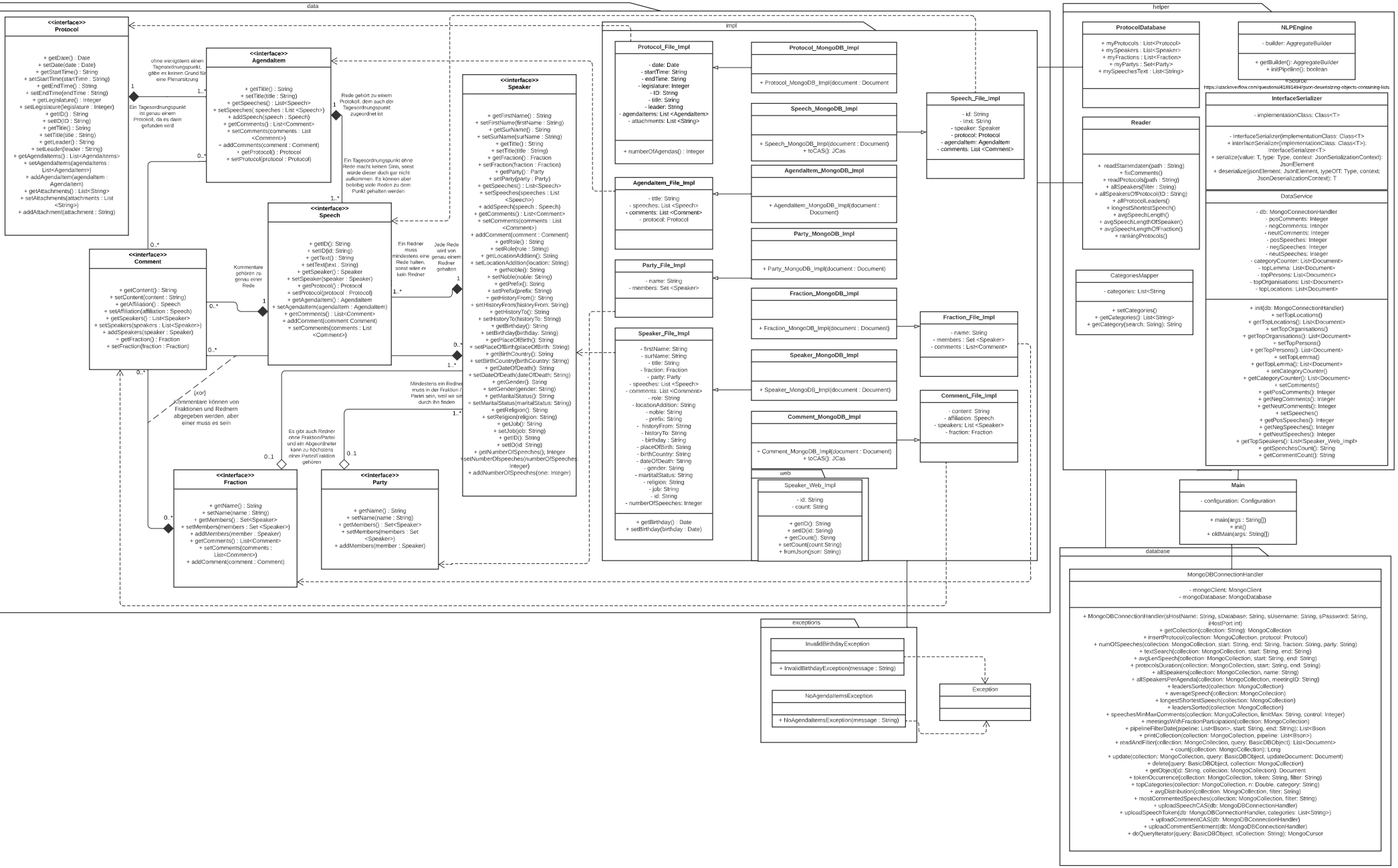
data
<<interface>>
Protocol
+ getDate() : Date
+ setDate(date : Date)
+ getStartTime() : String
+ setStartTime(startTime : String)
+ getEndTime() : String
+ setEndTime(endTime : String)
+ getLegislature() : Integer
+ setLegislature(legislature : Integer)
+ getID() : String
+ setID(ID : String)
+ getTitle() : String
+ setTitle(title : String)
+ getLeader() : String
+ setLeader(leader : String)
+ getAgendaItems() : List<AgendaItems>
+ setAgendaItems(agendaItems : List<AgendaItem>)
+ addAgendaItem(agendaItem : AgendaItem)
+ getAttachments() : List<String>
+ setAttachments(attachments : List <String>)
+ addAttachment(attachment : String)
ohne wenigstens einen Tagesordnungspunkt, gäbe es keinen Grund für eine Plenarsitzung
1
1..*
Ein Tagesordnungspunkt ist genau einem Protokoll, da es darin gefunden wird
<<interface>>
AgendaItem
+ getTitle() : String
+ setTitle(title : String)
+ getSpeeches() : List<Speech>
+ setSpeeches(speeches : List <Speech>)
+ addSpeech(speech : Speech)
+ getComments() : List<Comment>
+ setComments(comments : List <Comment>)
+ addComments(comment : Comment)
+ getProtocol() : Protocol
+ setProtocol(protocol : Protocol)
Rede gehört zu einem Protokoll, dem auch der Tagesordnungspunkt zugeordnet ist
1
Ein Tagesordnungspunkt ohne Rede macht keinen Sinn, sonst würde dieser doch gar nicht aufkommen. Es können aber beliebig viele Reden zu dem Punkt gehalten werden
0..*
1..*
<<interface>>
Speech
+ getID(): String
+ setID(id: String)
+ getText() : String
+ setText(text : String)
+ getSpeaker() : Speaker
+ setSpeaker(speaker : Speaker)
+ getProtocol() : Protocol
+ setProtocol(protocol : Protocol)
+ getAgendaItem() : AgendaItem
+ setAgendaItem(agendaItem : AgendaItem)
+ getComments() : List<Comment>
+ addComment(comment Comment)
+ setComments(comments : List <Comment>)
Ein Redner muss mindestens eine Rede halten, sonst wäre er kein Redner
Jede Rede wird von genau einem Redner gehalten
1
1..*
<<interface>>
Comment
+ getContent() : String
+ setContent(content : String)
+ getAffiliation() : Speech
+ setAffiliation(affiliation : Speech)
+ getSpeakers() : List<Speaker>
+ setSpeakers(speakers : List<Speaker>)
+ addSpeakers(speaker : Speaker)
+ getFraction() : Fraction
+ setFraction(fraction : Fraction)
Kommentare gehören zu genau einer Rede
0..*
1
0..*
0..*
0..*
{xor}
Kommentare können von Fraktionen und Rednern abgegeben werden, aber einer muss es sein
<<interface>>
Speaker
+ getFirstName() : String
+ setFirstName(firstName : String)
+ getSurName() : String
+ setSurName(surName : String)
+ getTitle() : String
+ setTitle(title : String)
+ getFraction() : Fraction
+ setFraction(fraction : Fraction)
+ getParty() : Party
+ setParty(party : Party)
+ getSpeeches() : List<Speech>
+ setSpeeches(speeches : List <Speech>)
+ addSpeech(speech : Speech)
+ getComments() : List<Comment>
+ setComments(comments : List <Comment>)
+ addComment(comment : Comment)
+ getRole() : String
+ setRole(role : String)
+ getLocationAddition(): String
+ setLocationAddition(location: String)
+ getNoble(): String
+ setNoble(noble: String)
+ getPrefix(): String
+ setPrefix(prefix: String)
+ getHistoryFrom(): String
+ setHistoryFrom(historyFrom: String)
+ getHistoryTo(): String
+ setHistoryTo(historyTo: String)
+ getBirthday(): String
+ setBirthday(birthday: String)
+ getPlaceOfBirth(): String
+ setPlaceOfBirth(placeOfBirth: String)
+ getBirthCountry(): String
+ setBirthCountry(birthCountry: String)
+ getDateOfDeath(): String
+ setDateOfDeath(dateOfDeath: String)
+ getGender(): String
+ setGender(gender: String)
+ getMaritalStatus(): String
+ setMaritalStatus(maritalStatus: String)
+ getReligion(): String
+ setReligion(religion: String)
+ getJob(): String
+ setJob(job: String)
+ getID(): String
+ setID(id: String)
+ getNumberOfSpeeches(): Integer
+ setNumberOfspeeches(numberOfSpeeches: Integer)
+ addNumberOfSpeeches(one: Integer)
0..*
1..*
Mindestens ein Redner muss in der Fraktion / Partei sein, weil wir sie durch ihn finden
1..*
Es gibt auch Redner ohne Fraktion/Partei und ein Abgeordneter kann zu höchstens einer Partei/Fraktion gehören
0..1
0..1
<<interface>>
Fraction
+ getName() : String
+ setName(name : String)
+ getMembers() : Set<Speaker>
+ setMembers(members : Set <Speaker>)
+ addMembers(member : Speaker)
+ getComments() : List<Comment>
+ setComments(comments : List<Comment>)
+ addComment(comment : Comment)
0..*
<<interface>>
Party
+ getName() : String
+ setName(name : String)
+ getMembers() : Set<Speaker>
+ setMembers(members : Set <Speaker>)
+ addMembers(member : Speaker)
impl
Protocol_File_Impl
- date: Date
- startTime: String
- endTime: String
- legislature: Integer
- ID: String
- title: String
- leader: String
- agendaItems: List <AgendaItem>
- attachments: List <String>
+ numberOfAgendas() : Integer
Protocol_MongoDB_Impl
+ Protocol_MongoDB_Impl(document : Document
Speech_MongoDB_Impl
+ Speech_MongoDB_Impl(document : Document)
+ toCAS(): JCas
Speech_File_Impl
- id: String
- text: String
- speaker: Speaker
- protocol: Protocol
- agendaItem: AgendaItem
- comments: List <Comment>
AgendaItem_File_Impl
- title: String
- speeches: List <Speech>
- comments: List <Comment>
- protocol: Protocol
AgendaItem_MongoDB_Impl
+ AgendaItem_MongoDB_Impl(document : Document)
Party_MongoDB_Impl
+ Party_MongoDB_Impl(document : Document)
Party_File_Impl
- name: String
- members: Set <Speaker>
Fraction_MongoDB_Impl
+ Fraction_MongoDB_Impl(document : Document)
Fraction_File_Impl
- name: String
- members : Set <Speaker>
- comments : List<Comment>
Speaker_File_Impl
- firstName: String
- surName: String
- title: String
- fraction: Fraction
- party: Party
- speeches: List <Speech>
- comments: List <Comment>
- role: String
- locationAddition: String
- noble: String
- prefix: String
- historyFrom: String
- historyTo: String
- birthday : String
- placeOfBirth: String
- birthCountry: String
- dateOfDeath: String
- gender: String
- maritalStatus: String
- religion: String
- job: String
- id: String
- numberOfSpeeches: Integer
+ getBirthday() : Date
+ setBirthday(birthday : Date)
Speaker_MongoDB_Impl
+ Speaker_MongoDB_Impl(document : Document)
Comment_File_impl
- content: String
- affiliation: Speech
- speakers: List <Speaker>
- fraction: Fraction
Comment_MongoDB_Impl
+ Comment_MongoDB_Impl(document : Document)
+ toCAS(): JCas
web
Speaker_Web_Impl
- id: String
- count: String
+ getID(): String
+ setID(id: String)
+ getCount(): String
+ setCount(count:String)
+ fromJson(json: String)
helper
ProtocolDatabase
+ myProtocols : List<Protocol>
+ mySpeakers : List<Speaker>
+ myFractions : List<Fraction>
+ myPartys : Set<Party>
+ mySpeechesText : List<String>
NLPEngine
- builder: AggregateBuilder
+ getBuilder(): AggregateBuilder
+ initPipeline(): boolean
Resource:
https://stackoverflow.com/questions/41891494/json-deserializing-objects-containing-lists
InterfaceSerializer
- implementationClass: Class<T>
- InterfaceSerializer(implementationClass: Class<T>
+ interfaceSerializer(implementationClass: Class<T>): InterfaceSerializer<T>
+ serialize(value: T, type: Type, context: JsonSerializationContext): JsonElement
+ deserialize(jsonElement: JsonElement, typeOfT: Type, context: JsonDeserializationContext): T
Reader
+ readSammdaten(path : String)
+ fixComments()
+ readProtocols(path : String)
+ allSpeakers(filter : String)
+ allSpeakersOfProtocol(ID : String)
+ allProtocolLeaders()
+ longestShortestSpeech()
+ avgSpeechLength()
+ avgSpeechLengthOfSpeaker()
+ avgSpeechLengthOfFraction()
+ rankingProtocols()
DataService
- db: MongoConnectionHandler
- posComments: Integer
- negComments: Integer
- neutComments: Integer
- posSpeeches: Integer
- negSpeeches: Integer
- neutSpeeches: Integer
- categoryCounter: List<Document>
- topLemma: List<Document>
- topPersons: List<Document>
- topOrganisations: List<Document>
- topLocations: List<Document>
+ init(db: MongoConnectionHandler)
+ setTopLocations()
+ getTopLocations(): List<Document>
+ setTopOrganisations()
+ getTopOrganisations(): List<Document>
+ setTopPersons()
+ getTopPersons(): List<Document>
+ setTopLemma()
+ getTopLemma(): List<Document>
+ setCategoryCounter()
+ getCategoryCounter(): List<Document>
+ setComments()
+ getPosComments(): Integer
+ getNegComments(): Integer
+ getNeutComments(): Integer
+ setSpeeches()
+ getPosSpeeches(): Integer
+ getNegSpeeches(): Integer
+ getNeutSpeeches(): Integer
+ getTopSpeakers(): List<Speaker_Web_Impl>
+ getSpeechesCount(): String
+ getCommentCount(): String
CategoriesMapper
- categories: List<String
+ setCategories()
+ getCategories(): List<String>
+ getCategory(search: String): String
Main
- configuration: Configuration
+ main(args : String[])
+ init()
+ oldMain(args: String[])
database
MongoDBConnectionHandler
- mongoClient: MongoClient
- mongoDatabase: MongoDatabase
+ MongoDBConnectionHandler(sHostName: String, sDatabase: String, sUsername: String, sPassword: String, iHostPort int)
+ getCollection(collection: String): MongoCollection
+ insertProtocol(collection: MongoCollection, protocol: Protocol)
+ numOfSpeeches(collection: MongoCollection, start: String, end: String, fraction: String, party: String)
+ textSearch(collection: MongoCollection, start: String, end: String)
+ avgLenSpeech(collection: MongoCollection, start: String, end: String)
+ protocolsDuration(collection: MongoCollection, start: String, end: String)
+ allSpeakers(collection: MongoCollection, name: String)
+ allSpeakersPerAgenda(collection: MongoCollection, meetingID: String)
+ leadersSorted(collection: MongoCollection)
+ averageSpeech(collection: MongoCollection)
+ longestShortestSpeech(collection: MongoCollection)
+ leadersSorted(collection: MongoCollection)
+ speechesMinMaxComments(collection: MongoCollection, limitMax: String, control: Integer)
+ meetingsWithFractionParticipation(collection: MongoCollection)
+ pipelineFilterDate(pipeline: List<Bson>, start: String, end: String): List<Bson
+ printCollection(collection: MongoCollection, pipeline: List<Bson>)
+ readAndFilter(collection: MongoCollection, query: BasicDBObject): List<Document>
+ count(collection: MongoCollection): Long
+ update(collection: MongoCollection, query: BasicDBObject, updateDocument: Document)
+ delete(query: BasicDBObject, collection: MongoCollection)
+ getObject(id: String, collection: MongoCollection): Document
+ tokenOccurrence(collection: MongoCollection, token: String, filter: String)
+ topCategories(collection: MongoCollection, n: Double, category: String)
+ avgDistribution(collection: MongoCollection, filter: String)
+ mostCommentedSpeeches(collection: MongoCollection, filter: String)
+ uploadSpeechCAS(db: MongoDBConnectionHandler)
+ uploadSpeechToken(db: MongoDBConnectionHandler, categories: List<String>)
+ uploadCommentCAS(db: MongoDBConnectionHandler)
+ uploadCommentSentiment(db: MongoDBConnectionHandler)
+ doQueryIterator(query: BasicDBObject, sCollection: String): MongoCursor
exceptions
InvalidBirthdayException
+ InvalidBirthdayException(message : String)
NoAgendaItemsException
+ NoAgendaItemsException(message : String)
Exception