UMLClass Diagram // Backend // Parliament_Browser_9_4

parliament_browser_9_4.main.java

data

impl

>>Interface<<
Speech

+ getID():String
+ getSpeakerID(): String
+ getText(): String
+ getDate(): LocalTime

>>Class<<
Speech_Impl

- speechDoc: Document
- _id: String
- speakerID: String
- text: String
-date: LocalDate

0...n

>>Interface<<
Protocol

+ getID(): String
+ getElectionperiod(): int
+ getDate(): LocalDate
+ getProtocolNumber(): int
+ getBeginTime(): LocalTime
+ getEndTime(): LocalTime
+ getSessionLeaders(): List<String>
+ getAgendaItemIDs(): List<String>

>>Class<<
Protocol_Impl

- protocolDoc: Document
- _id: String
- beginTime: LocalTime
- endTime: LocalTime
- date:LocalDate
- electionPeriod:int
- protocolNumber: int
- sessionLeaders: ArrayList<String>
- agendaItems: ArrayList<String>

0...n

0...n

>>Interface<<
Person

+ getID(): String
+ getFirstName(): String
+ getLastName(): String
+ getRole(): String
+ getTitle(): String
+ getFraction(): String
+ getParty(): String
+ getPlace(): String
+ getGender(): String
+ getBirthDate(): String
+ getDeathDate(): String
+ setPfpMetadata(String data): void
+ getPfpMetadata(): String
+ setPfpUrl(String url): void
+ getPfpUrl(): String

>>Class<<
Person_Impl

- personDoc: Document
- _id: String
- firstName: String
- lastName: String
- role: String
- title: String
- fraction: String
- party: String
- electionPeriods: ArrayList<String>
- place: String
- birthDate: String
- bithPlace: String
- deathDate: String
- gender: String

0...n

>>Interface<<
Comment

+ getID(): String
+ getSpeechID(): String
+ getSpeakerID(): String
+ getCommentator(): String
+ getFractions(): List<String>
+ getText(): String
+ getDate(): LocalDate

>>Class<<
Comment_Impl

- commentDoc: Document
- _id: String
- speechID: String
- speakerID: String
- commentatorID: String
- text: String
- date: LocalDate
- fractions: ArrayList<String>

0...n

Extends

Extends

>>Interface<<
AgendaItem

+ getID(): String
+ getDate(): LocalDate
+ getSpeechIDs(): ArrayList<String>
+ getSubject(): String

>>Class<<
AgendaItem_Impl

- ID: String
- titel: String
- Protokoll: Protokoll
- Kommentare: Set<Kommentar>
- protokollHandler: ProtokollHander
- document: Document
- reden: Set<Rede>

+ loadData(Node n): void

0...n

>>Class<<
Poll_Impl

- _id: int
- date: LocalDate
- spd: int{}
- cxu: int{}
- b90: int{}
- fdp: int{}
- afd: int{}
- linke: int{}
- independent: int{}

0..n

>>Interface<<
Poll

+ toJson(): String

Extends

tex

>>Class<<
AgendaItem_TeX

+ AgendaItem_TeX(Document)
+ toTeX(): String
- aiDoc: Document
- mdbh: MongoDBHandler

1...n

instantiates

1

>>Class<<
Protocol_TeX

+ Protocol_TeX(Document)
+ toTeX(): String
- protDoc: Document
- mdbh: MongoDBHandler
- targetDirectory: String

1

instantiates

1...n

>>Class<<
Speech_TeX

+ toTeX(): String
+ nlpTableTex(String): String
+ toTeX(String): String

- mdbh: MognoDBHandler
- DECIMAL_FORMAT: DecimalF

>>Class<<
GoodWindowsExec

+ main(String[]): void

>>Class<<
StreamGobbler

+ StreamGobbler(InputStream, String)
+ run(): void
+ is: InputStream
+type: String

instantiates

1

>>Class<<
LaTeXHandler

+ createTEX(String): String
+ createPDF(String: void
+ createTexFile(String): void

- targetDir: String
- mdbh: MongoDBHandler

utility

>>Class<<
EditorProtocolParser

+ parseEditorProtocol(JSONObject, boolean): String
+ parseEditorAgendaItem(JSONObject, boolean): String
+ parseEditorSpeech(JSONObject, boolean): String
+ parseEditorPerson(JSONObject, boolean): String
- validateProtocolID(String): int{}
+ getEditorProtocolFromDB(String): JSONObject
+ getEditorAgendaFromDB(String): JSONObject
+ getEditorSpeechFromDB(String): JSONObject
+ getEditorPersonFromDB(String): JSONObject
- mongoDBHandler: MongoDBHandler
- uo: UpdateOptions
- uima: UIMAPerformer
- protocolReqs: List<String>
- agendaReqs: List<String>
- speechReqs: List<String>
- personReqs: List<String>

>>Class<<
User

+ getID(): String
+ getRank(): String

- id: String
- rank: String

Use

uima

>>Class<<
MongoSentence

+ getStartPos(): int
+ getEndPos(): int
+ getSentiment

- startPos: int
- endPos: int
- sentiment: double

1...*

1...*

>>Class<<
MongoToken

+ getStartPos(): int
+ getEndPos(): int
+ getLemmaValue(): String
+ getPOS(): String

- startPos: int
- endPos: int
- lemmaValue: String
- pos: String

>>Class<<
MongoNamedEntity

+ getStartPos(): int
+ getEndPos(): int
+ getEntityType(): String
+ getCoveredText(): String

- startPos: int
- endPos: int
- entityType: String
- coveredText: String

1...*

1...*

>>Class<<
ProcessedSpeech

+ toSpeechJson(): String
+ toSpeechCasJson(): String
+ toSpeechTokensJson(): String

- _id: String
- speakerID: String
- text: String
- date: LocaDate
- fullCas: String
- sentiment: double
- mainTopic: String
- tokens: List<MongoToken>
- sentences: List<MongoSentence>
- namedEntities: List<MongoNamedEntity>

0..n

Use

>>Class<<
TimeHelper

+ convertToISOdate(String date): LocalDate
+ convertToISOtime(String time): LocalTime
+ durationBetweenTimesInMinutes(Temporal begin, Temporal end): long

- DATE_FORMAT_INPUT: DateTimeFormatter
- CLOCK_FORMAT_INPUT: DateTimeFormatter

instantiates and uploads to Database

>>Class<<
XMLPersonParser

+ personParse(): void
+ getNodesByName(Node n, String s): List<Node>
+ getSingleNode(Node n, String s): Node

- persons: ArrayList<Person>
- protocols: ArrayList<Protocol>
- agendaItems: ArrayList<AgendaItem>

Use

>>Class<<
XMLProtocolParser

+ speechParse(): void
- getAllElements(NodeList): Map<String, String>
- getElementList(Element, String): List<Element>
- getChildElementList(Element): List<Element>
+ addToSpeechMap(String..., Boolean, MongoDBHandler, int): int
+ addToCommentMap(String..., Integer, LocalDate, ...): void

- speechMap: Map<String, Speech>
- commentMap: Map<String, Comment>
- dbf: DocumentBuilderFactory

1

instantiates and uploads to Database

Use

Use

Use

Creates Instances and inserts into Database

1

>>Class<<
MongoDBHandler

+ getCollection(String col): MongoCollection<Document>
+ collectionExists(String col): boolean
+ getDocument(String col, String id): void
+ createCollection(String col): void
+ addDocument(Document doc,String col): void
+ addDocuments(List<Documents> docs):void
+ updateDocument(Document doc, String col, String id): boolean
+ deleteDocument(String id, String col): void
+ insertPersons(List<Person> person): void
+ insertPerson(Person person): void
+ insertProtocols(List<Protocols> protocols): void
+ insertProtocol(Protocol protocol): void
+ insertSpeech(Speech s, String fullCas, List<MongoTokens>, ...)
+ insertAgendaItems(List<AgendaItems>): void
+ insertComment(Comment c, double sentiment)
+ applyDateFilterToAggregation(String fil, String f, List<Bson> pipel
+ aggregateIterate(String s, Bson... pipeline): Iterable<Document>
+ getText(String col, String id): String
+ addCAS(String col, String id, String cas): void
+ checkIfHasNonEmptyField(String col, String id, String field)
+ checkIfDocumentExists(String col, String id)
+ getSpeechesBySpeakerCount(): void
+ getTokenCount(): void
+ getPersonEntities(): void
+ getOrganisationEntities(): void
+ getLocationEntities(): void

- db: MongoDatabase
- gson: Gson

Use

>>Class<<
UIMAPerformer

+ getJCas(String text): JCas
+ getFullCas(JCas jcas): String
+ getTokens(JCas jcas): List<MongoToken>
+ getSentences(JCas jcas): List<MongoSentence>
+ getNamedEntities(JCas jcas) List<MongoNamedEntity>
+ getMainTopic(JCas jcas): String
+ getAverageSentiment(JCas jcas): double
+ serializeFromDB(String col, String id): void
+ serializeToDB(String col, String id, String text): void
- validateQuery(String col, String id): void
- serializeData(String col, String id, String text): void
- generateAnalysisEngine(): AnalysisEngine
- generateDDCCategories(): String{}

- mongoDBHandler: MongoDBHandler
- analysisEngine: AnalysisEngine
- ddcCategories: String {}

Use

Use

Use

Use

>>Class<<
SparkHandler

+ main(String[] args): void
+init(MongoDBHandler): void
+ responseJson(String, Object): JSON
+ openInDefaultBrowser(String): void
+ openInDefaultBrowser(): void

- mongoDBHandler: MongoDBHandler
- cfg: Configuration
- epParser: EditorProtocolParser
- frontendPath: String
- getHome: TemplateViewRoute
- postHome: Route
- getLaTeX: Route
- getLaTeXString: Route
- postLatex: Route
- getProtokollEditor: TemplateViewRoute
- postProtokollEditorInsert: Route
- postProtokollEditorExtract: Route
- getDashboard: TemplateViewRoute
- getChartUpdates: Route
- getReden: TempalteViewRoute
- getSpeechVis: Route
- getSpeechIDs: Route
- getSpeechNetwork: TemplateviewRoute
- getCommentNetwork: TemplateviewRoute
- getSpeechTopicNetwork: TemplateviewRoute
- getLoginSite: TemplateViewRoute
- postChangePassword: Route
- postDeleteUser: Route
- postLogout: Route
- postRegister: Route
- postLogin: Route
- postEditUser: Route

>>Class<<
PollScraper

+ getAllPolls(MongoDBHandler): List<Poll>
+ getOnePoll(int): Poll

- noPollCounter: int

1

instantiates and uploads to Databse

>>Class<<
PictureScraper

+ producePictureUrl(String firstName, String lastName): String{}

>>Class<<
Scraper

+ downloadAllXMLs(): void
+ downloadTenXML(String url): int
+ getFileName(String url): String
+ checkSite(String url)

Use

Use

Interacts with

throws

throws

throws

throws

exceptions

>>Class<<
WrongInputException

+ WorngInputException()
+ WorngInputException(String m)

>>Class<<
NoPollException

+ NoPollExcetion()
+ NoPollException(String m)

>>Class<<
EditorFormattingException

+ EditorFormattingException()
+ EditorFormattingException(String m)

MongoDB

Handles interactions with

bundestag.de
// Website
//Database

Handles Connection with Frontend

Frontend

CSS

Public

JS

FTL