App

App
+ main(String[] arg): void

<< Interface >>
ITeacher
+ doExercise(Exercise): T

Teacher
+ doExercise(Exercise): T

<< Interface >>
ITeacher
+ mark(input: Exercise): void

Teacher
+ mark(input: Exercise): void

Scene
- IBotUI: IUIHelper
- Scene: IUIHelper
+ Scene (mongoDb: MongoDb): void
+ start(): void

Handlers

Handler
+ process(exchange: HttpServerExchange, context: Map<String,Object>, header: String): void

Frontend

TrayHandler
- context: Map<String,Object>
- tray: Tray
- header: String
+ GetHandle(): void
+ put(Connect: Hashtable<String,String>): void
+ get(Connect: Hashtable<String,String>): void

AccountHandler
- mongoDb: MongoDb
- accountCollection: AccountCollection
+ AccountHandler(mongoDb: MongoDb): void
+ writeAccountBalance(HttpServerExchange exchange): void
+ writeAccount(HttpServerExchange exchange): void
+ deleteAll(HttpServerExchange exchange): void
+ find(HttpServerExchange exchange): void
+ login(HttpServerExchange exchange): void
+ checkUser(HttpServerExchange exchange): void
+ findUsers(HttpServerExchange exchange): void
+ getUsersBy(HttpServerExchange exchange): void

AdminHandler
- mongoDb: MongoDb
- adminCollector: AdminCollector
+ AdminHandler(mongoDb: MongoDb): void
+ ping(HttpServerExchange exchange): void
+ log(HttpServerExchange exchange): void
+ addAdmin(HttpServerExchange exchange): void
+ deleteUser(HttpServerExchange exchange): void

CourseHandler
- mongoDb: MongoDb
- CourseCollector: CourseCollector
+ addCourse(HttpServerExchange exchange): void
+ updateCourse(HttpServerExchange exchange): void

LectureHandler
- mongoDb: MongoDb
- LectureCollector: LectureCollector
+ log(HttpServerExchange exchange): void
+ logoutUser(HttpServerExchange exchange): void
+ getAllLecture(HttpServerExchange exchange): void

PathHandler
- pathCollector: PathCollector
- accountCollector: AccountCollector
- mongoDb: mongoDb
+ PathHandler(mongoDb: MongoDb): void
+ addPath(HttpServerExchange exchange): void
+ addPathToUser(HttpServerExchange exchange): void
+ deletePath(HttpServerExchange exchange): void
+ updatePath(HttpServerExchange exchange): void
+ updateOrder(HttpServerExchange exchange): void
+ updatePathTo(HttpServerExchange, s:String): void
+ getPath(HttpServerExchange exchange): void

User Handler
- mongoDb: MongoDb
- pathCollector: PathCollector
+ UserHandler(mongoDb: MongoDb)
+ handleLogout(HttpServerExchange exchange)

User RelationHandler
- mongoDb: MongoDb
+ UserRelationHandler(mongoDb: MongoDb)
+ handleLogout(HttpServerExchange exchange)

Models

VueModel
- context: Map<String,Object>
+ VueModel(): void
+ addUserInfo(List<String> labels, String type, String): void
+ addLogout(Logout Body, Exception Body): void
+ put(Long Body, ex: Object): void
+ get Context(): Map<String,Object>

Controllers

AccountController
- accountManager: AccountManager
- mongoDb: MongoDb
+ AccountController(mongoDb: MongoDb): void
+ readAll(): List<String>
+ log(): void
+ deleteAll(): void
+ isBetween(ent: String, ent2: String): boolean
+ follow(ent: String, ent2: String): void
+ unfollow(ent: String, ent2: String): void
+ search(keyword: String, limit: int): List<User, String>
+ getFollowers(ent: String): List<User, String>
+ getFollowing(ent: String): List<User, String>
+ getFollowersFrom(search: List<User, String>): List<User, String>
+ isAdmin(username: String): boolean
+ isBand(username: String): boolean

AdminController
- accountManager: AccountManager
- pathManager: PathManager
- courseManager: CourseManager
+ AdminController(mongoDb: MongoDb): void
+ ping(ex: Long): void
+ log(ex: Long): void
+ addAdmin(ex: Long): void
+ deleteUser(ex: Long): void

CourseController
- courseCollector: CourseCollector
+ CourseController(mongoDb: MongoDb): void
+ addCourse(pathId: UUID, course: String, author: String): void
+ getCourseInfo(pathId: UUID): List<User, String>

LectureController
- courseManager: CourseManager
+ lecture(courseId: String, pathId: String): String
+ log(courseId: String, pathId: String): String
+ logoff(courseId: String, pathId: String): String

PathController
- accountManager: AccountManager
- pathManager: PathManager
+ PathController(mongoDb: MongoDb): void
+ addPath(title: String, author: String, author: String): void
+ getPathByBody(Author: String): List<User, String>
+ deletePath(pathId: UUID): void
+ getPath(pathId: UUID): Map<String, String>
+ getStrength(courseId: String): List<User, String>

Use Cases

MongoDb
- accountManager: AccountManager
- pathManager: PathManager
- courseManager: CourseManager
- connection: String
+ MongoDb(accountManager: AccountManager, pathManager: PathManager, courseManager: CourseManager): void
+ getAccountManager(): AccountManager
+ get PathManager(): PathManager
+ get CourseManager(): CourseManager
+ getConnection(): String
+ getConnection(connection: String): void
+ getConnectionStatus(): boolean
+ ping(): void

Data Mapper

DataMapper
- model: List<Map<String,String>>
+ getAttr(object: T, attribute: String): String
+ read(): void
+ addAttr(row: T, attribute: String[]): void
+ deleteAttr(row: T, attribute: String[]): void
+ editAttr(row: T, Double, attribute: String[]): void
+ getModel(): List<Map<String,String>>
+ getTableMap(key: T, attribute: String[]): Map<String,String>

Entities

AccountManager
- accountMap: Hashtable<String, Account>
- reader: IReader
- writer: IWriter
+ AccountManager(reader: IReader, writer: IWriter): void
+ follow(follower: String, followee: String): void
+ unfollow(follower: String, followee: String): void
+ getFollowersOf(username: String): Hashtable<String>
+ getFollowingOf(username: String): Hashtable<String>
+ isFollower(username: String): boolean
+ isAdmin(username: String): boolean
+ getUser(username: String): Account
+ addUser(username: String, account: Account): void
+ deleteUser(username: String): void
+ ban(username: String): void
+ unban(username: String): void
+ topUp(username: String, pretend: String): void
+ getAllUsers(username: String): void
+ getUserByTag(username: String): List<UserDetailsDto>
+ CreatePathway(username: String, pretend: String): void
+ deletePathway(username: String)
+ save(): void

PartManager
- parts: Hashtable<UUID,Part>
- reader: IReader
- writer: IWriter
- partSorter: IPartSorter
+ PartManager(reader: IReader, writer: IWriter, sorter): void
- createPart(title: String, content: String, author: String): Part
+ getPartsByBody(username: String): Map<int,Part>
+ deleteUserPathwayBody(username: String): void
+ addPart(title: String, content: String, author: String): UUID
+ deletePart(id: UUID): void
+ getPart(id: UUID): void
+ save(): void
+ setPartSorter(IPartSorter partSorter): void

CourseManager
- courses: Hashtable<UUID,Course>
- reader: IReader
- writer: IWriter
- courseSorter: ICourseSorter
+ CourseManager(reader: IReader, writer: IWriter): void
- create Course(pathId: UUID, content: String, author: String): Course
+ getCourseByBody(username: String): Map<int,Course>
+ getCourseByID(pathId: UUID): Map<int,Course>
+ deleteCourseByBody(username: String): void
+ addCourse(pathId: UUID, content: String, author: String): UUID
+ deleteCourse(id: UUID): void
+ getCourse(id: UUID): Course
+ setCourseSorter(CourseSorter: ICourseSorter): void
+ save(): void

Interfaces

<< Interface >>
IReader
+ read(path: String): String

FileReader
+ read(path: String): String

Account
- username: String
- hashPassword: String
- tray: List<UserDetailsDto>
- isBanned: boolean
- isAdmin: boolean
- Follows: Hashtable<String>
- Followers: Hashtable<String>
+ Account(username: String, hashPassword: String): void
+ follow(user: String): void
+ unfollow(user: String): void
+ addFollower(user: String): void
+ removeFollower(user: String): void
+ getFollows(): Hashtable<String>
+ getFollowers(): Hashtable<String>
+ getUsernames(): String
+ getUsernameLike(username: String): void
+ getHashPassword(): String
+ setHashPassword(): void
+ getIsBanned(): boolean
+ getIsAdmin(): boolean
+ getIsCreatorAdmin(): boolean
+ ban(): boolean
+ unban: boolean

Part
- title: String
- content: String
- author: String
- hashtags: List<Hashtag>
- id: UUID
+ Part(title: String, content: String, author: String): void
+ getTitle(): String
+ getContent(): String
+ getAuthor(): String
+ getID(): UUID
+ setTitle(title: String): void
+ setContent(content: String): void
+ getTemplate(): List<Hashtag>

Course
- pathID: UUID
- content: String
- author: String
- hashtags: List<Hashtag>
- id: UUID
+ Course(pathID: UUID, content: String, author: String): void
+ get Content(): String
+ getAuthor(): String
+ getID(): UUID
+ getTemplate(): List<Hashtag>
+ getPathID(): UUID

Gateways

<< Interface >>
IPartSorter
+ sort(parts: List<Part>): List<Part>

<< Interface >>
ITitleSorter
+ sort(parts: List<Part>): List<Part>

<< Interface >>
ICourseSorter
+ sort(parts: List<Course>): List<Course>

PartSorter
- stringMatcher: StringMatcher
+ getSearch(List<Part>, keywords: String): List<P>

<<create>>
<<use>>

PartComparator
+ compare(P1: Part, P2: Part): int

StringMatcher
+ editDistance(word: String, word2: String)

TitleSorter

<<create>>
<<use>>

PartTimeComparator
+ compare(P1: Part, P2: Part): int

CourseSorter

<<create>>
<<use>>

CourseTimeComparator
+ compare(C1: Course, C2: Course): int

Data Generator

CommandRuleGenerator
+ main(String[] arg): void

PathRuleGenerator
+ main(String[] arg): void

UserRuleGenerator
+ main(String[] arg): void

Exceptions

PartNotBoundException
+ PartNotBoundException(errMessage: String): void

UserNotFoundException
+ UserNotFoundException(errMessage: String): void

UserNotAllowedToUseException
+ UserNotAllowedToUseException(errMessage: String): void

TagNotBoundException
+ TagNotBoundException(errMessage: String): void

UserNotAdminException
+ UserNotAdminException(errMessage: String): void

UserNotFollowedException
+ UserNotFollowedException(errMessage: String): void

TagNotAdmonException
+ TagNotAdmonException(errMessage: String): void

UserNotLoginException
+ UserNotLoginException(errMessage: String): void