THU Chatbot

Generated by Doxygen 1.10.0

Chapter 1

API Routes for ThuBOT

1.1 API Documentation

The maven dependency `springdoc-openapi` automatically generates the API documentation under `/v3/api-docs` and also provides the Swagger UI under `/swagger-ui/index.html`

1.2 API Guidelines

For the sake of consistency and clean code practices, the routes will start with `/api`, followed by the object controller name, followed by the operation (example: `/api/users/create`) Please follow common HTTP verb rules where applicable (example: a `GET` route should only be used to retrieve information, etc.)

1.3 API authentication

Each route will be protected with one of two permissions:

- USER
- SYSTEM

More information in the permissions_model.MD

Chapter 2

Permissions model for ThuBOT routes

There are 2 permission levels:

- USER -> accessible by users with role "user"
- SYSTEM -> accessible by users with role "system"

The authentication will be done with the `Authentication` header and will take a bearer token of the user.

When a permissions error occurs (not enough permissions, or auth header missing), a `403 FORBIDDEN` will be returned.

Chapter 3

tree_vision

1. Make the tree statically expanded

Have the viewbox expand portionally to the tree size, so that each node has the correct distance between each other

1. Functionality to drag around the viewbox, by holding down the left click, and zooming by using the scroll wheel.
2. Add in sidebox as a submit form from the database/view site, where we can create a node, delete a node and modify a node

---------- Web Scraper Vision ----------

1. Find a webscraper tool (https://jsoup.org/cookbook/introduction/parsing-a-document)
2. Strips down entire html source code and extracts only the text. Removes all tags adn comments and only keeps the text that is in betweeen the tags for example, in between tr, p etc.

1. Some way to store the stripped down information for each revelant page that we use for our chatbot

Chapter 4

How to contribute to the project in GitHub

With this repository there is a GitHub project attached, which allows for a sprint board linked with issues (tasks). You will be assigned tasks, or choose one yourself.

In order to contribute code, you have to:

- create a new branch using the naming scheme `feature/tasknr` or `bugfix/bugid`
- do the task
- create a pull request

Chapter 5

THUBot First Increment

5.1 Current state of the project

Implemented:

- user system with roles and privileges
- frontend for login, chatbot (more or less Work In Progress)
- essential classes and models (DialogNode, Response, OpenAIService)
- logic for dialog flow
- staging environment (web and db)
- dev db environment (individual users and databases)
- unit tests
- basic set of dialog nodes and responses

Work in progress:

- complete code coverage in unit tests
- fully functioning dialog node matching
- registration page for users
- extend dialog node repository

Planned:

- multi language support (EN, DE)
- backend for manipulating dialog nodes and chat parameters
- end-to-end tests
- light / dark mode for all pages
- impressum, privacy policy, contact pages

5.2 Screenshots

Login page

Chatbot page

Chapter 6

THUBot Second Increment

6.1 Current state of the project

Implemented:

- unit tests for around 20% of the codebase
- functioning dialog node matching
- login and registration page for users
- visual backend for adding, deleting and modifying dialog nodes
- filter chain with access denied handler (for backend pages)

Work in progress:

- fully functioning frontend (JS calls to the backend routes)
- end-to-end tests
- impressum, privacy policy, contact pages

Planned:

- multi language support (EN, DE)
- light / dark mode for all pages
- basic voice access to the bot (still a prototype)

6.2 Screenshots

Visual backend for adding, modifying and deleting nodes

Access denied handlers (Security filter chain)

Chapter 7

playground_openai

Example request to OpenAI:

```
curl  https://api.openai.com/v1/chat/completions \ -H "Content-Type:  application/json"
\ -H "Authorization:  Bearer $OPENAI_API_KEY" \ -d '{ "model":  "gpt-3.↩
5-turbo", "messages":  [ { "role":  "system", "content":  "You are extracting
keywords from user queries passed to a chat bot for a university.  You give
the relevant keywords, separated by comma and nothing else.  Your first
word will be ONLY one of those:  WHERE,HOW,WHEN,WHAT based on what the question
wants" }, { "role":  "user", "content":  "when does the campus at prittwitzstrasse
close on sunday?" }, { "role":  "assistant", "content":  "WHEN, campus,
prittwitzstrasse, close, Sunday" } ], "temperature":  0.24, "max_tokens"↩
:  256, "top_p":  1, "frequency_penalty":  0, "presence_penalty":  0 }
```

Chapter 8

THU BOT Team Roles

Eugen: Software Architect, Dev Simon: Dev, Prompt Engineer Yassine: QA, Dev Ria: UI/UX, Prompt Engineer Josh: Dev, Infrastructure Osama: Dev, Scrum Master Bobby: Databases Firas: UI/UX, Translation

Chapter 9

THU BOT Technical Constraints

- Availability: Uptime should be 99%
- Maintainability: The code base should be easily updatable.
- Performance: Response time under 5 seconds.
- Scalability: It is enough that the system is horizontally expandable
- Reliablity: The chat bot should be able to handle 25 concurrent requests.
- Compatibility: The system should run on every device that has an active internet connection and runs a modern web browser (V8)
- Localization: The system should support European units (KM, DDMMYYYY, etc.) and the English and German languages as defaults.
- Usability: The program should be self explainatory and use a standard layout compatible with accessibility features already present on most devices.

Chapter 10

THU BOT Software Project (CTS5)

So far, there is only a backend API implemented using Spring Boot.

10.1 Main architecture

The important concepts are:

- DialogNode
- Chatbot
- UserInput

Each DialogNode is one of three types (`ROOT`, `INTERMEDIARY` or `QUESTION`), has one parent and `n` children.

A practical example:

Therefore, the user will be prompted with the next possibilities in the dialogue only. When using natural language input, the OpenAI API is called with only the choices of the next level. Example: Asking a user whether they are a student or prospective student -> Send input along with the choices STUDENT, PROSPECTIVE STUDENT.

This helps in reducing the amount of tokens used with the API. More info on tokens can be found in the `OpenAI documentation`.

The following sequence diagram shows the interaction with OpenAI's API:

10.2 Requirements

Technology Stack (so far):

- Java 17
- Spring Boot with the following gradle dependencies:
- - spring-boot-starter-data-jpa
- - spring-boot-starter-web
- - mysql-connector-j
- MySQL Server

10.3 Running the backend

Use `maven` in order to build the project. The root directory has a `pom.xml` file which has the necessary configuration in order to build the app along with its dependencies. Open the terminal and run `mvn spring-boot:run`

10.3.1 IntelIJ Idea

There should be built in support for everything out of the box. Just open the repository and there should be a pop-up asking to run the gradle build.

10.3.2 VSCode

Install the `Extenstion Pack for Java` extension pack in VS Code. This will bring all the necessary tools. Afterward, reload VS Code and open the repository. There should be a pop-up asking if the project should use `maven` or `gradle`. Choose `maven`. The build should be running in the background, and you can check its status by clicking th icon in the bottom right corner. (look up `vscode maven integration` for more info and debugging)

Before running the code, you have to adjust the configuration in the `application.properties` file inside `src/main/resources`. Copy the file `application.properties.example` to `application.properties` and change the DB connection settings and OpenAPI API key.

NEVER PUSH THIS FILE TO GIT! It is in the .gitignore, and it should stay that way. Do not modify the example file, otherwise credentials WILL get leaked and your OpenAI account WILL be terminated.

This assumes you are running a MySQL server on localhost port 3306 (default for MySQL) and you have a db called THUBOT_DB and a user DB_USERNAME with DB_PASSWORD that has access to that database. The easiest option is to download MySQL Server for Win, Mac or Linux and run the installer. It will have a GUI configurator that is easy to use.

10.4 PlantUML and Diagrams

Useful docs for PlantUML: `https://plantuml.com/`

Create your diagrams (if needed) in the `docs/` folder as .puml files, and afterwards export them as PNG files to `docs/generated/`

Chapter 11

Namespace Index

11.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 12

Hierarchical Index

12.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 13

Class Index

13.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 14

File Index

14.1 File List

Here is a list of all files with brief descriptions:

Chapter 15

Namespace Documentation

15.1 Package com.zegline.thubot

Packages

- package core

Classes

- class ThuBotApplication

  Main application class for the ThuBot chatbot.

15.2 Package com.zegline.thubot.core

Packages

- package config
- package controller
- package model
- package repository

Classes

- class ApiApplicationTests

  Test class for the main application context.
- class DataLoader

15.3 Package com.zegline.thubot.core.config

Classes

- class SecurityConfig

  Configures the security settings including security filters, user details, and password encoding.
- class SetupDataLoader

  This class initiates some database content on application startup.

15.4 Package com.zegline.thubot.core.controller

Classes

- class DialogNodeController

  Provides REST endpoints to manage DialogNodes.
- class GUIController

  Handles the requests related to GUI display and interactions.
- class UserController

  Controller class for User-related actions.
- class UserInputController

  Controller to manage user input related actions.

15.5 Package com.zegline.thubot.core.model

Packages

- package security

Classes

- class DialogNode

  Represents a node within a dialog conversation.
- class DialogNodeTests

  Test class for the DialogNode.
- class DialogNodeToResponse

  Represents a link between a DialogNode and a Response entity.
- class Response

  Entity representing a response in the system.

15.6 Package com.zegline.thubot.core.model.security

Classes

- class Privilege

  Entity to represent a specific privilege.
- class Role

  Entity to represent a specific role.
- class User

  Entity representing a user.

15.7 Package com.zegline.thubot.core.repository

Classes

- interface DialogNodeRepository

  Interface providing CRUD operations and custom queries for DialogNode entities.
- class DialogNodeRepositoryTests

  Test class for the DialogNodeRepository.
- interface DialogNodeResponseRepository

  Interface providing CRUD operations and custom queries for DialogNodeToResponse entities.
- interface PrivilegeRepository

  Interface providing CRUD operations and custom queries for Privilege entities.
- interface ResponseRepository

  Repository interface providing CRUD operations for Response entities.
- interface RoleRepository

  Repository interface providing CRUD operations and a method to find Role by name.
- interface UserRepository

  Repository interface for User entities providing CRUD operations and additional methods.

15.8 Package com.zegline.thubot.core.service.dialogNodeMatch

Classes

- class DialogNodeMatch

  Service class for matching user input to dialog nodes.
- class DialogNodeMatchTests

15.9 Package com.zegline.thubot.core.service.openai

Classes

- class OpenAIService

  Service class for OpenAI API interaction.

15.10 Package com.zegline.thubot.core.service.user

Classes

- class ThuUserDetailService

  Service class for loading user-specific data needed for authentication.
- class ThuUserPrincipal

  Implements UserDetails interface for User authentication.

15.11 Package com.zegline.thubot.core.utils.generator

Classes

- class QuestionIdGenerator

  Custom ID generator for Hibernate entities.

Chapter 16

Class Documentation

16.1 com.zegline.thubot.core.ApiApplicationTests Class Reference

Test class for the main application context.

Package Functions

- void contextLoads ()

16.1.1 Detailed Description

Test class for the main application context.

This class contains a test to load and initialize the application context.

Definition at line 23 of file apiApplicationTests.java.

16.1.2 Member Function Documentation

16.1.2.1 contextLoads()

```
void com.zegline.thubot.core.ApiApplicationTests.contextLoads ( )  [package]
```

Tests the application context loading.

This test case ensures that the application context loads successfully without any issues.

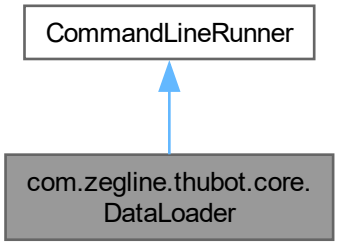
Definition at line 31 of file apiApplicationTests.java.

The documentation for this class was generated from the following file:

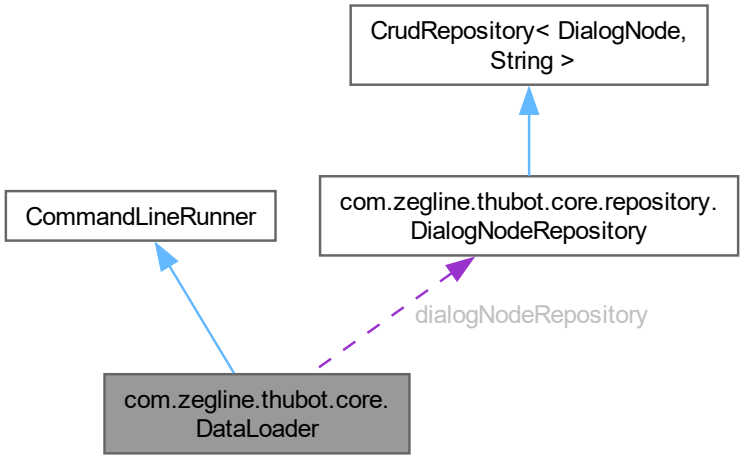
- thuBOT/src/test/java/com/zegline/thubot/core/apiApplicationTests.java

16.2 com.zegline.thubot.core.DataLoader Class Reference

Inheritance diagram for com.zegline.thubot.core.DataLoader:



Collaboration diagram for com.zegline.thubot.core.DataLoader:



Public Member Functions

- void run (String... args) throws Exception

Private Attributes

- DialogNodeRepository dialogNodeRepository

16.2.1 Detailed Description

Definition at line 13 of file DataLoader.java.

16.2.2 Member Function Documentation

16.2.2.1 run()

```
void com.zegline.thubot.core.DataLoader.run (
            String...  args ) throws Exception
```

Definition at line 19 of file DataLoader.java.

16.2.3 Member Data Documentation

16.2.3.1 dialogNodeRepository

```
DialogNodeRepository com.zegline.thubot.core.DataLoader.dialogNodeRepository  [private]
```

Definition at line 16 of file DataLoader.java.

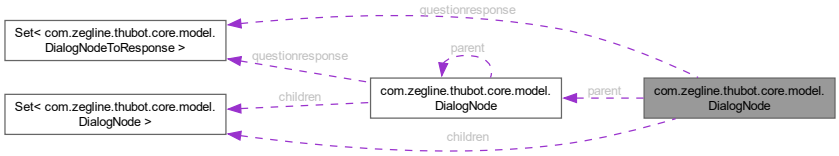
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/DataLoader.java

16.3 com.zegline.thubot.core.model.DialogNode Class Reference

Represents a node within a dialog conversation.

Collaboration diagram for com.zegline.thubot.core.model.DialogNode:



Public Member Functions

- DialogNode (String dialogText, String msgText)
- DialogNode addChild (DialogNode c)
- DialogNode addChildren (Set< DialogNode > nodes)
- String toString ()
- void setDialogText (String dialogText)
- void setMsgText (String msgText)
- boolean equals (Object obj)
- int hashCode ()

Package Attributes

- Set< DialogNodeToResponse > questionresponse

Private Attributes

- String id
- String dialogText
- String msgText
- DialogNode parent
- Set< DialogNode > children = new HashSet<>()

16.3.1 Detailed Description

Represents a node within a dialog conversation.

A dialog node is an entity that contains text for both the dialog (question) and the message (response). It is part of a larger conversation flow and can have relationships to other nodes

Definition at line 38 of file DialogNode.java.

16.3.2 Constructor & Destructor Documentation

16.3.2.1 DialogNode()

```
com.zegline.thubot.core.model.DialogNode.DialogNode (
            String dialogText,
            String msgText )
```

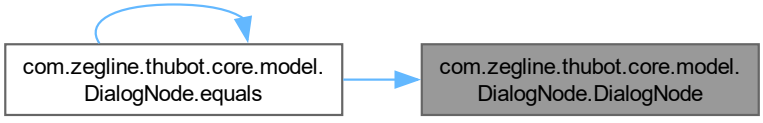
Constructor for the DialogNode.

Parameters

dialogText	String that the dialog node will contain.
msgText	String that is printed out after the DialogNode's text.

Definition at line 74 of file DialogNode.java.

Here is the caller graph for this function:



16.3.3 Member Function Documentation

16.3.3.1 addChild()

```
DialogNode com.zegline.thubot.core.model.DialogNode.addChild (
            DialogNode c )
```

Adds a child to the current DialogNode.

Parameters

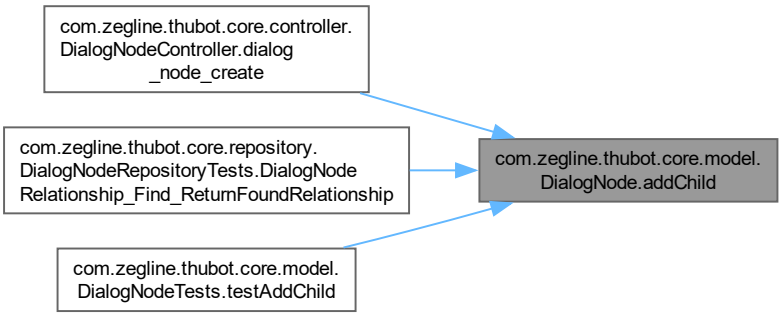
child	DialogNode to be added to the children list of the DialogNode.

Returns

The updated DialogNode with the new child.

Definition at line 85 of file DialogNode.java.

Here is the caller graph for this function:



16.3.3.2 addChildren()

```
DialogNode com.zegline.thubot.core.model.DialogNode.addChildren (
            Set< DialogNode > nodes )
```

Adds multiple children to the current DialogNode.

Parameters

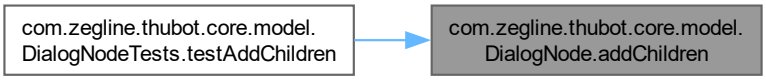
nodes	A set of children DialogNodes to be added to the children list of the DialogNode.

Returns

The updated DialogNode with the new children.

Definition at line 97 of file DialogNode.java.

Here is the caller graph for this function:



16.3.3.3 equals()

```
boolean com.zegline.thubot.core.model.DialogNode.equals (
            Object obj )
```

Checks if the current DialogNode is equal to another DialogNode

Parameters

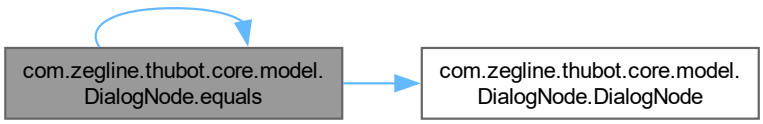
obj	Object to be compared for equality

Returns

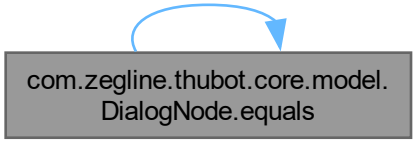
boolean representing the result of the equality check

Definition at line 136 of file DialogNode.java.

Here is the call graph for this function:



Here is the caller graph for this function:



16.3.3.4 hashCode()

```
int com.zegline.thubot.core.model.DialogNode.hashCode ( )
```

Calculates and returns the hash code for the DialogNode

Returns

int representing the hash code of the DialogNode

Definition at line 148 of file DialogNode.java.

16.3.3.5 setDialogText()

```
void com.zegline.thubot.core.model.DialogNode.setDialogText (
            String dialogText )
```

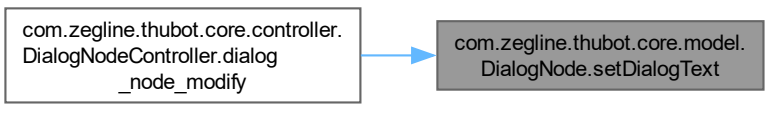
Sets the text for the DialogNode

Parameters

dialogText	The text to be set

Definition at line 118 of file DialogNode.java.

Here is the caller graph for this function:



16.3.3.6 setMsgText()

```
void com.zegline.thubot.core.model.DialogNode.setMsgText (
            String msgText )
```

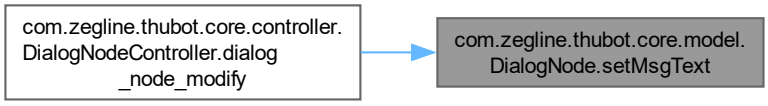
Sets the message text for the DialogNode

Parameters

msgText	The message text to be set

Definition at line 126 of file DialogNode.java.

Here is the caller graph for this function:



16.3.3.7 toString()

```
String com.zegline.thubot.core.model.DialogNode.toString ( )
```

Returns a string representation of the DialogNode.

Returns

String The dialog text that the node contains enclosed in "<Dialog> ".

Definition at line 110 of file DialogNode.java.

16.3.4 Member Data Documentation

16.3.4.1 children

```
Set<DialogNode> com.zegline.thubot.core.model.DialogNode.children = new HashSet<>()  [private]
```

Definition at line 66 of file DialogNode.java.

16.3.4.2 dialogText

```
String com.zegline.thubot.core.model.DialogNode.dialogText  [private]
```

Definition at line 50 of file DialogNode.java.

16.3.4.3 id

```
String com.zegline.thubot.core.model.DialogNode.id  [private]
```

Definition at line 47 of file DialogNode.java.

16.3.4.4 msgText

```
String com.zegline.thubot.core.model.DialogNode.msgText  [private]
```

Definition at line 54 of file DialogNode.java.

16.3.4.5 parent

```
DialogNode com.zegline.thubot.core.model.DialogNode.parent  [private]
```

Definition at line 62 of file DialogNode.java.

16.3.4.6 questionresponse

```
Set<DialogNodeToResponse> com.zegline.thubot.core.model.DialogNode.questionresponse  [package]
```

Definition at line 57 of file DialogNode.java.

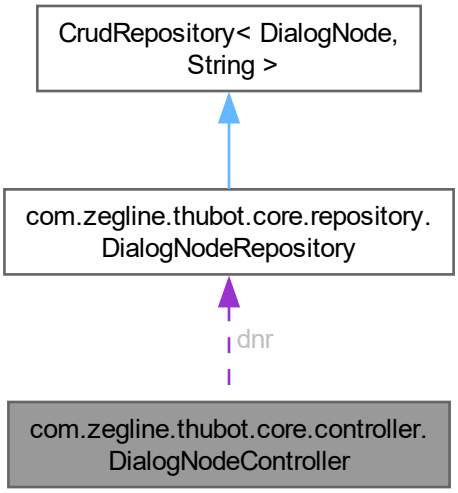
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/model/DialogNode.java

16.4 com.zegline.thubot.core.controller.DialogNodeController Class Reference

Provides REST endpoints to manage DialogNodes.

Collaboration diagram for com.zegline.thubot.core.controller.DialogNodeController:



Public Member Functions

- DialogNode dialog_node_create (@RequestBody Map< String, String > body)
- DialogNode dialog_node_modify (@RequestBody Map< String, String > body)
- DialogNode dialog_node_delete (@RequestBody Map< String, String > body)
- Set< DialogNode > get (@RequestBody(required=false) Map< String, String > body)

Package Attributes

- DialogNodeRepository dnr

16.4.1 Detailed Description

Provides REST endpoints to manage DialogNodes.

This controller class defines the routes for creating, modifying, deleting, and retrieving DialogNodes. All endpoints of this controller are under the "/api/dialognode" route.

Definition at line 32 of file DialogNodeController.java.

16.4.2 Member Function Documentation

16.4.2.1 dialog_node_create()

```
DialogNode com.zegline.thubot.core.controller.DialogNodeController.dialog_node_create (
            @RequestBody Map< String, String > body )
```

Creates a new DialogNode based on the provided data in the request body.

Parameters

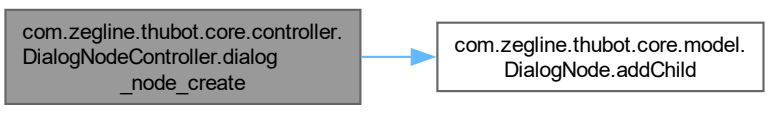
body	A map containing dialogNodeText and msgText to create the DialogNode.

Returns

The created DialogNode.

Definition at line 44 of file DialogNodeController.java.

Here is the call graph for this function:



16.4.2.2 dialog_node_delete()

```
DialogNode com.zegline.thubot.core.controller.DialogNodeController.dialog_node_delete (
            @RequestBody Map< String, String > body )
```

Deletes a DialogNode based on the provided ID in the request body.

Parameters

body	A map containing the ID of the DialogNode to delete.

Returns

The parent of the deleted DialogNode.

Exceptions

ResponseStatusException	if the specified DialogNode is not found, if it has children, or the specified DialogNode is the root.

Definition at line 109 of file DialogNodeController.java.

16.4.2.3 dialog_node_modify()

```
DialogNode com.zegline.thubot.core.controller.DialogNodeController.dialog_node_modify (
            @RequestBody Map< String, String > body )
```

Modifies an existing DialogNode based on the provided data in the request body.

Parameters

body	A map containing dialogNodeId and the new dialogNodeText, msgText, and parentNodeId.

Returns

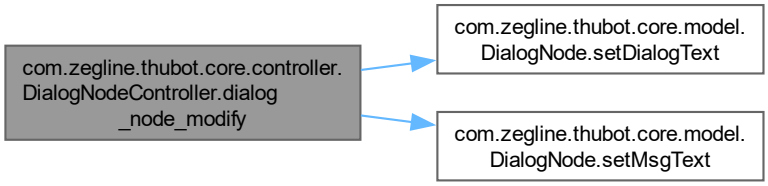
The modified DialogNode.

Exceptions

ResponseStatusException	if the specified DialogNode is not found.

Definition at line 72 of file DialogNodeController.java.

Here is the call graph for this function:



16.4.2.4 get()

```
Set< DialogNode > com.zegline.thubot.core.controller.DialogNodeController.get (
            @RequestBody(required=false) Map< String, String > body )
```

Retrieves DialogNode(s) based on the provided parameters or returns all DialogNodes if no parameters are specified.

Parameters

body	A map containing parameters for filtering DialogNodes (optional).

Returns

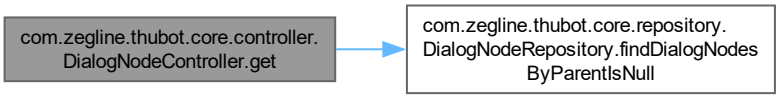
Set of DialogNodes matching the provided criteria or all DialogNodes if no specific parameters are provided.

Exceptions

ResponseStatusException	If the provided ID is empty or if the DialogNode with the specified ID is not found.

Definition at line 148 of file DialogNodeController.java.

Here is the call graph for this function:



16.4.3 Member Data Documentation

16.4.3.1 dnr

```
DialogNodeRepository com.zegline.thubot.core.controller.DialogNodeController.dnr  [package]
```

Definition at line 35 of file DialogNodeController.java.

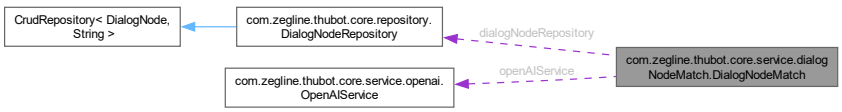
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/controller/DialogNodeController.java

16.5 com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch Class Reference

Service class for matching user input to dialog nodes.

Collaboration diagram for com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch:



Public Member Functions

- DialogNode getResponseNode (String userInput)

Static Public Member Functions

- static List< DialogNode > getNodesRecursively (DialogNode root, int recurseLevel)

Private Member Functions

- List< String > getAnswers (List< DialogNode > nodes)
- DialogNode matchNodeToInput (String input)

Static Private Member Functions

- static void getNodesRecursivelyHelper (DialogNode node, int recurseLevel, List< DialogNode > result)

Private Attributes

- DialogNodeRepository dialogNodeRepository
- OpenAIService openAIService

16.5.1 Detailed Description

Service class for matching user input to dialog nodes.

Offers methods to match user input against dialog nodes from a repository or generate responses using OpenAI's services. It encapsulates the logic for the dialog node matching process.

Definition at line 29 of file DialogNodeMatch.java.

16.5.2 Member Function Documentation

16.5.2.1 getAnswers()

```
List< String > com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch.getAnswers (
            List< DialogNode > nodes )  [private]
```

This method retrieves all answers from a list of dialog nodes.

Parameters

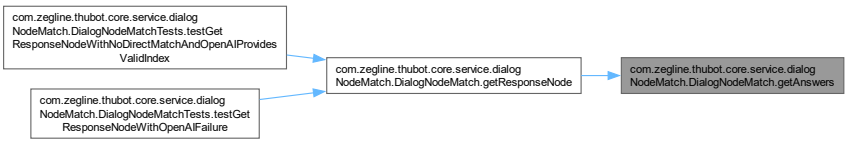
nodes	A list of dialog nodes from which to extract responses

Returns

A list of answers, represented by the values of the msgText fields of the dialog nodes.

Definition at line 77 of file DialogNodeMatch.java.

Here is the caller graph for this function:



16.5.2.2 getNodesRecursively()

```
static List< DialogNode > com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch.↩
getNodesRecursively (
            DialogNode root,
            int recurseLevel )  [static]
```

This method recursively retrieves all dialog nodes starting from a root node.

Parameters

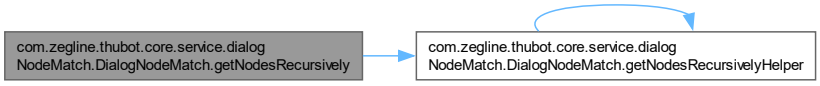
root	The root dialog node from which to start the search.
recurseLevel	The maximum depth of the search.

Returns

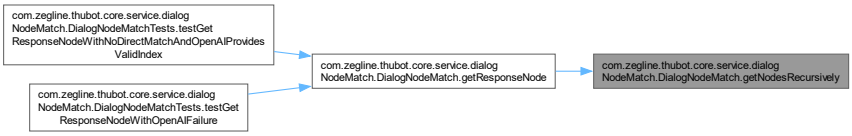
A list of all dialog nodes reachable from the root node, up to the specified search depth.

Definition at line 92 of file DialogNodeMatch.java.

Here is the call graph for this function:



Here is the caller graph for this function:



16.5.2.3 getNodesRecursivelyHelper()

```
static void com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch.getNodesRecursively↩
Helper (
            DialogNode node,
            int recurseLevel,
            List< DialogNode > result )  [static], [private]
```

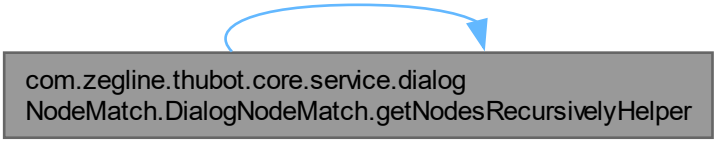
This method is a helper for the getNodesRecursively method, executing the actual recursion.

Parameters

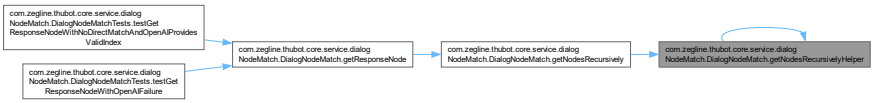
node	The node from which to start the search.
recurseLevel	The remaining search depth.
result	A running list of dialog nodes found so far.

Definition at line 105 of file DialogNodeMatch.java.

Here is the call graph for this function:



Here is the caller graph for this function:



16.5.2.4 getResponseNode()

```
DialogNode com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch.getResponseNode (
            String userInput )
```

Matches user input with responses in the databases.

Parameters

userInput	User input string, could be the prompt or natural language.

Returns

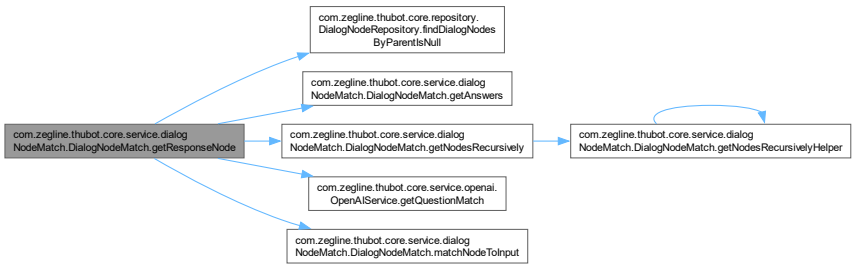
The fetched answer or a default message if no suitable response is found.

Exceptions

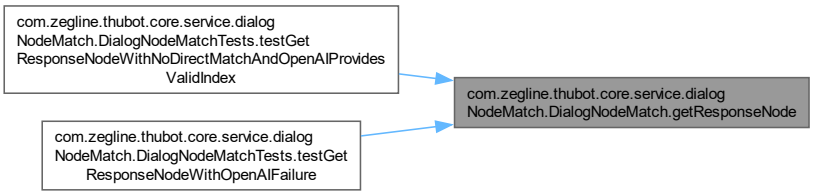
Exception	If there's an error parsing the matched question number returned by OpenAI.

Definition at line 45 of file DialogNodeMatch.java.

Here is the call graph for this function:



Here is the caller graph for this function:



16.5.2.5 matchNodeToInput()

```
DialogNode com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch.matchNodeToInput (
            String input )  [private]
```

Placeholder method to implement database matching logic.

Parameters

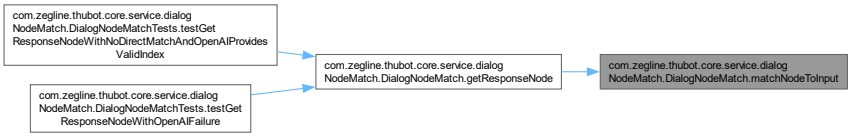
input	input string to match

Returns

Matched node if found, otherwise null.

Definition at line 125 of file DialogNodeMatch.java.

Here is the caller graph for this function:



16.5.3 Member Data Documentation

16.5.3.1 dialogNodeRepository

```
DialogNodeRepository com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch.dialog↩
NodeRepository  [private]
```

Definition at line 32 of file DialogNodeMatch.java.

16.5.3.2 openAIService

```
OpenAIService com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch.openAIService
[private]
```

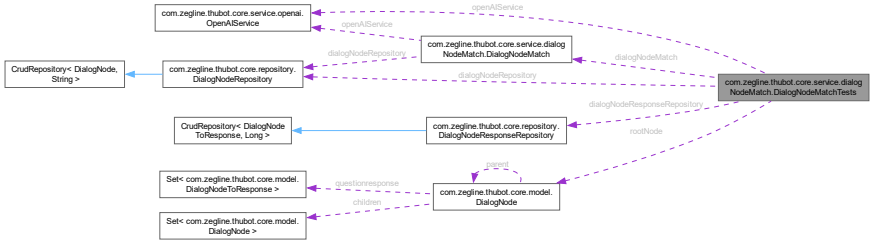
Definition at line 35 of file DialogNodeMatch.java.

The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/service/dialogNodeMatch/DialogNodeMatch.java

16.6 com.zegline.thubot.core.service.dialogNodeMatch.DialogNode↩ MatchTests Class Reference

Collaboration diagram for com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests:



Public Member Functions

- void setup ()
- void testGetResponseNodeWithNoDirectMatchAndOpenAIProvidesValidIndex ()
- void testGetResponseNodeWithOpenAIFailure ()

Private Attributes

- DialogNodeRepository dialogNodeRepository
- DialogNodeResponseRepository dialogNodeResponseRepository
- OpenAIService openAIService
- DialogNodeMatch dialogNodeMatch
- DialogNode rootNode

16.6.1 Detailed Description

Definition at line 30 of file DialogNodeMatchTests.java.

16.6.2 Member Function Documentation

16.6.2.1 setup()

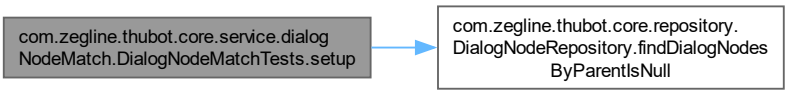
```
void com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests.setup ( )
```

Setup method that runs before each test

This method sets up the necessary components for each test

Definition at line 52 of file DialogNodeMatchTests.java.

Here is the call graph for this function:



16.6.2.2 testGetResponseNodeWithNoDirectMatchAndOpenAIProvidesValidIndex()

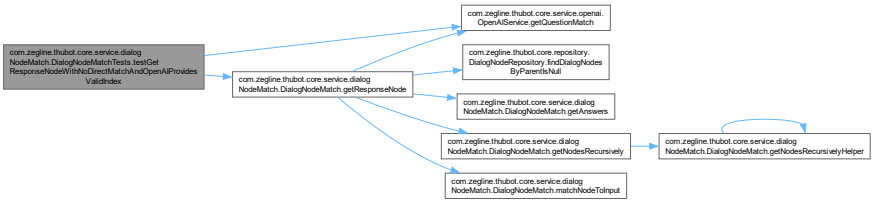
```
void com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests.testGetResponse↩
NodeWithNoDirectMatchAndOpenAIProvidesValidIndex ( )
```

Test method for getResponseNode when there's no direct match but OpenAI provides a valid index

This method tests a scenario where there's no direct match for the input, but the OpenAI service provides a valid index for a DialogNode match. It verifies that the right DialogNode is returned.

Definition at line 64 of file DialogNodeMatchTests.java.

Here is the call graph for this function:



16.6.2.3 testGetResponseNodeWithOpenAIFailure()

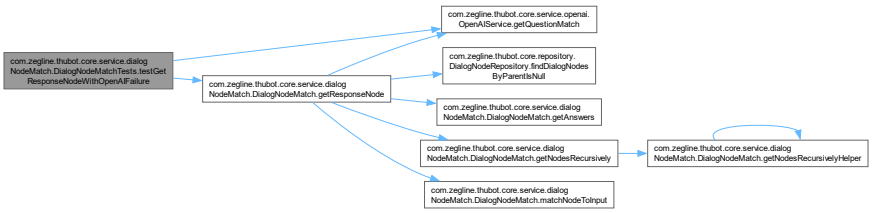
```
void com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests.testGetResponse↩
NodeWithOpenAIFailure ( )
```

Test method for getResponseNode when OpenAI fails to provide a match

This method tests a scenario where the OpenAI service fails to provide a valid match for the input. It verifies that a default DialogNode instance is returned when no match is found.

Definition at line 79 of file DialogNodeMatchTests.java.

Here is the call graph for this function:



16.6.3 Member Data Documentation

16.6.3.1 dialogNodeMatch

```
DialogNodeMatch com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests.dialog↩
NodeMatch  [private]
```

Definition at line 42 of file DialogNodeMatchTests.java.

16.6.3.2 dialogNodeRepository

```
DialogNodeRepository com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests.↩
dialogNodeRepository  [private]
```

Definition at line 33 of file DialogNodeMatchTests.java.

16.6.3.3 dialogNodeResponseRepository

```
DialogNodeResponseRepository com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch↩
Tests.dialogNodeResponseRepository  [private]
```

Definition at line 36 of file DialogNodeMatchTests.java.

16.6.3.4 openAIService

```
OpenAIService com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests.open↩
AIService  [private]
```

Definition at line 39 of file DialogNodeMatchTests.java.

16.6.3.5 rootNode

```
DialogNode com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests.rootNode  [private]
```

Definition at line 44 of file DialogNodeMatchTests.java.

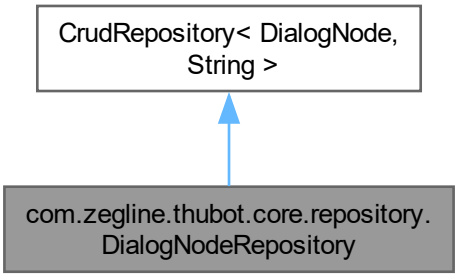
The documentation for this class was generated from the following file:

- thuBOT/src/test/java/com/zegline/thubot/core/service/dialogNodeMatch/DialogNodeMatchTests.java

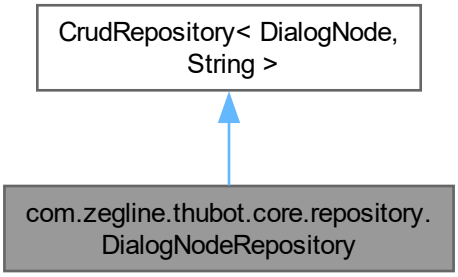
16.7 com.zegline.thubot.core.repository.DialogNodeRepository Interface Reference

Interface providing CRUD operations and custom queries for DialogNode entities.

Inheritance diagram for com.zegline.thubot.core.repository.DialogNodeRepository:



Collaboration diagram for com.zegline.thubot.core.repository.DialogNodeRepository:



Public Member Functions

- DialogNode findByChildren (DialogNode child)
- List< DialogNode > findDialogNodesByParentIsNull ()

16.7.1 Detailed Description

Interface providing CRUD operations and custom queries for DialogNode entities.

Definition at line 23 of file DialogNodeRepository.java.

16.7.2 Member Function Documentation

16.7.2.1 findByChildren()

```
DialogNode com.zegline.thubot.core.repository.DialogNodeRepository.findByChildren (
            DialogNode child )
```

Finds a parent DialogNode of the specified child node.

Parameters

child	The DialogNode to search for in the parent's children.

Returns

The parent DialogNode of the specified child, or null if no parent is found.

16.7.2.2 findDialogNodesByParentIsNull()

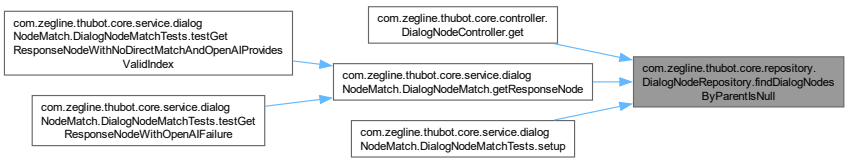
```
List< DialogNode > com.zegline.thubot.core.repository.DialogNodeRepository.findDialogNodesBy↩
ParentIsNull ( )
```

Finds all DialogNodes which have no parent node (root nodes).

Returns

List of DialogNode instances without a parent node.

Here is the caller graph for this function:



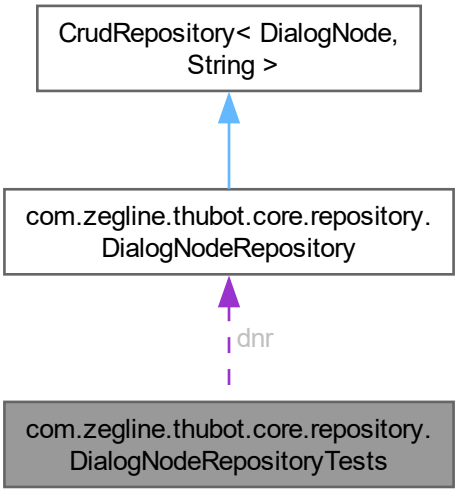
The documentation for this interface was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/repository/DialogNodeRepository.java

16.8 com.zegline.thubot.core.repository.DialogNodeRepositoryTests Class Reference

Test class for the DialogNodeRepository.

Collaboration diagram for com.zegline.thubot.core.repository.DialogNodeRepositoryTests:



Public Member Functions

- void DialogNode_Find_ReturnFoundDialogNode ()
- void DialogNodeRelationship_Find_ReturnFoundRelationship ()

Package Attributes

- DialogNodeRepository dnr

16.8.1 Detailed Description

Test class for the DialogNodeRepository.

Tests the DialogNodeRepository's ability to save and retrieve DialogNode instances, as well as the correct handling of relationships between Parent and Child nodes.

Definition at line 33 of file DialogNodeRepositoryTests.java.

16.8.2 Member Function Documentation

16.8.2.1 DialogNode_Find_ReturnFoundDialogNode()

```
void com.zegline.thubot.core.repository.DialogNodeRepositoryTests.DialogNode_Find_Return↩
FoundDialogNode ( )
```

Test method for saving a DialogNode and retrieving it.

This method saves a DialogNode to the database and then retrieves the saved DialogNode to verify that it has been persisted correctly.

Definition at line 45 of file DialogNodeRepositoryTests.java.

16.8.2.2 DialogNodeRelationship_Find_ReturnFoundRelationship()

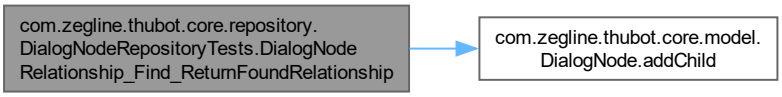
```
void com.zegline.thubot.core.repository.DialogNodeRepositoryTests.DialogNodeRelationship_↩
Find_ReturnFoundRelationship ( )
```

Test method for saving DialogNodes with a Parent-Child relationship and retrieving them.

This method saves two DialogNode objects (Parent and Child) to the database, and retrieves them to verify that the relationships between nodes has been persisted correctly.

Definition at line 64 of file DialogNodeRepositoryTests.java.

Here is the call graph for this function:



16.8.3 Member Data Documentation

16.8.3.1 dnr

```
DialogNodeRepository com.zegline.thubot.core.repository.DialogNodeRepositoryTests.dnr  [package]
```

Definition at line 36 of file DialogNodeRepositoryTests.java.

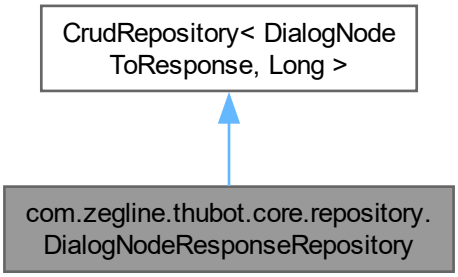
The documentation for this class was generated from the following file:

- thuBOT/src/test/java/com/zegline/thubot/core/repository/DialogNodeRepositoryTests.java

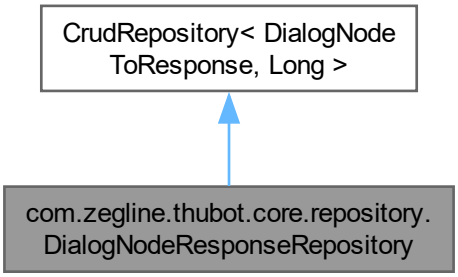
16.9 com.zegline.thubot.core.repository.DialogNodeResponseRepository Interface Reference

Interface providing CRUD operations and custom queries for DialogNodeToResponse entities.

Inheritance diagram for com.zegline.thubot.core.repository.DialogNodeResponseRepository:



Collaboration diagram for com.zegline.thubot.core.repository.DialogNodeResponseRepository:



Public Member Functions

- List< DialogNodeToResponse > findByDialogNode (DialogNode dialogNode)

16.9.1 Detailed Description

Interface providing CRUD operations and custom queries for DialogNodeToResponse entities.

Definition at line 23 of file DialogNodeResponseRepository.java.

16.9.2 Member Function Documentation

16.9.2.1 findByDialogNode()

```
List< DialogNodeToResponse > com.zegline.thubot.core.repository.DialogNodeResponseRepository.↩
findByDialogNode (
            DialogNode dialogNode )
```

Finds all DialogNodeToResponse associations for a specified DialogNode.

Parameters

dialogNode	The DialogNode to find associations for.

Returns

List of DialogNodeToResponse instances associated with the specified DialogNode.

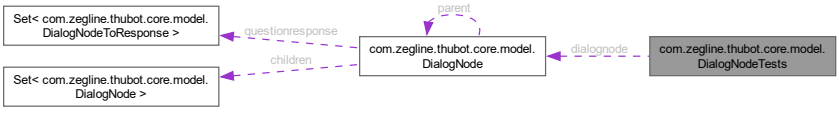
The documentation for this interface was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/repository/DialogNodeResponseRepository.java

16.10 com.zegline.thubot.core.model.DialogNodeTests Class Reference

Test class for the DialogNode.

Collaboration diagram for com.zegline.thubot.core.model.DialogNodeTests:



Public Member Functions

- void setup ()
- void testDialogNodeCreation ()
- void testAddChild ()
- void testAddChildren ()

Private Attributes

- DialogNode dialognode

16.10.1 Detailed Description

Test class for the DialogNode.

This class contains unit tests for the DialogNode entity. It tests the functionality of creating a DialogNode object and adding child nodes to a parent node.

Definition at line 33 of file DialogNodeTests.java.

16.10.2 Member Function Documentation

16.10.2.1 setup()

```
void com.zegline.thubot.core.model.DialogNodeTests.setup ( )
```

Setup method that runs before each test

This method sets up a DialogNode object to be used in each test

Definition at line 43 of file DialogNodeTests.java.

16.10.2.2 testAddChild()

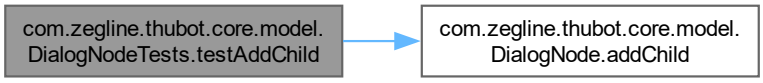
```
void com.zegline.thubot.core.model.DialogNodeTests.testAddChild ( )
```

Test method for adding a child node to a DialogNode

This method tests the method for adding a single child node to a DialogNode object

Definition at line 64 of file DialogNodeTests.java.

Here is the call graph for this function:



16.10.2.3 testAddChildren()

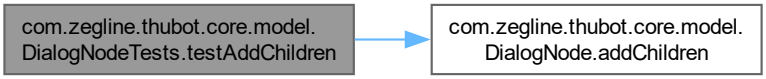
```
void com.zegline.thubot.core.model.DialogNodeTests.testAddChildren ( )
```

Test method for adding multiple children to a DialogNode

This method tests the method for adding multiple child nodes to a DialogNode object

Definition at line 78 of file DialogNodeTests.java.

Here is the call graph for this function:



16.10.2.4 testDialogNodeCreation()

```
void com.zegline.thubot.core.model.DialogNodeTests.testDialogNodeCreation ( )
```

Test method for DialogNode creation

This method tests the creation of a DialogNode object and its getters

Definition at line 53 of file DialogNodeTests.java.

16.10.3 Member Data Documentation

16.10.3.1 dialognode

```
DialogNode com.zegline.thubot.core.model.DialogNodeTests.dialognode  [private]
```

Definition at line 35 of file DialogNodeTests.java.

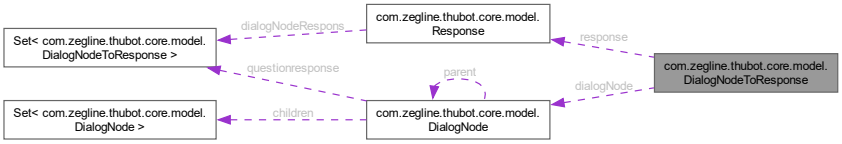
The documentation for this class was generated from the following file:

- thuBOT/src/test/java/com/zegline/thubot/core/model/DialogNodeTests.java

16.11 com.zegline.thubot.core.model.DialogNodeToResponse Class Reference

Represents a link between a DialogNode and a Response entity.

Collaboration diagram for com.zegline.thubot.core.model.DialogNodeToResponse:



Public Member Functions

- DialogNode getQuestion ()
- Response getResponse ()

Private Attributes

- long id
- DialogNode dialogNode
- Response response

16.11.1 Detailed Description

Represents a link between a DialogNode and a Response entity.

This entity serves as a join table in the database which holds the association between a dialog node and a response. Each DialogNode can have multiple responses associated with it, and each response can be associated with multiple dialog nodes

Definition at line 24 of file DialogNodeToResponse.java.

16.11.2 Member Function Documentation

16.11.2.1 getQuestion()

```
DialogNode com.zegline.thubot.core.model.DialogNodeToResponse.getQuestion ( )
```

Returns the DialogNode question associated with this DialogNode-Response relationship.

Returns

The DialogNode associated with this relationship.

Definition at line 42 of file DialogNodeToResponse.java.

16.11.2.2 getResponse()

```
Response com.zegline.thubot.core.model.DialogNodeToResponse.getResponse ( )
```

Returns the Response entity associated with this DialogNode-Response relationship.

Returns

The Response associated with this relationship.

Definition at line 51 of file DialogNodeToResponse.java.

16.11.3 Member Data Documentation

16.11.3.1 dialogNode

```
DialogNode com.zegline.thubot.core.model.DialogNodeToResponse.dialogNode  [private]
```

Definition at line 31 of file DialogNodeToResponse.java.

16.11.3.2 id

```
long com.zegline.thubot.core.model.DialogNodeToResponse.id  [private]
```

Definition at line 27 of file DialogNodeToResponse.java.

16.11.3.3 response

```
Response com.zegline.thubot.core.model.DialogNodeToResponse.response  [private]
```

Definition at line 35 of file DialogNodeToResponse.java.

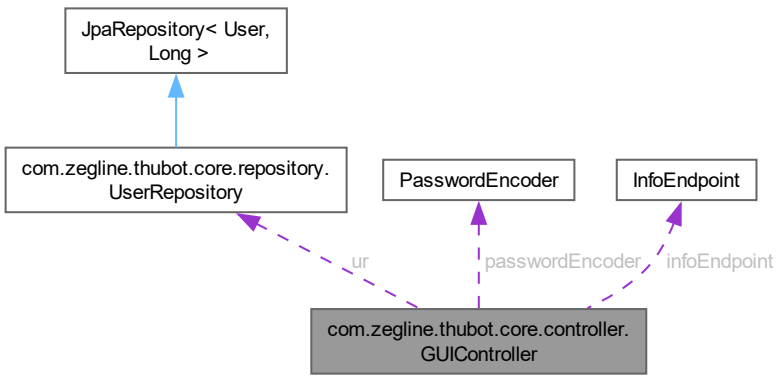
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/model/DialogNodeToResponse.java

16.12 com.zegline.thubot.core.controller.GUIController Class Reference

Handles the requests related to GUI display and interactions.

Collaboration diagram for com.zegline.thubot.core.controller.GUIController:



Public Member Functions

- String getIndex (Model model)
- String getDN ()
- String getDN1 ()
- String getDBEntry (Model model, @AuthenticationPrincipal UserDetails userDetails)
- String login (Model model)
- String showRegisterForm (Model model)
- String accessDeniedPage ()
- String registerUser (Model model, @RequestParam Map< String, String > body)

Private Attributes

- UserRepository ur
- PasswordEncoder passwordEncoder
- InfoEndpoint infoEndpoint

16.12.1 Detailed Description

Handles the requests related to GUI display and interactions.

Provides endpoints for rendering the GUI pages and populates the frontend views with data fetched from backend services like application actuator information. It also provides endpoints for user registration and login.

Definition at line 37 of file GUIController.java.

16.12.2 Member Function Documentation

16.12.2.1 accessDeniedPage()

```
String com.zegline.thubot.core.controller.GUIController.accessDeniedPage ( )
```

Serves an access-denied view.

Returns

Name of the view for the access-denied page.

Definition at line 128 of file GUIController.java.

16.12.2.2 getDBEntry()

```
String com.zegline.thubot.core.controller.GUIController.getDBEntry (
            Model model,
            @AuthenticationPrincipal UserDetails userDetails )
```

Serves the database form view.

Parameters

model	The model in which to place attributes for the view.
userDetails	The UserDetails object of the currently authenticated user.

Returns

Name of the view for the database form.

Definition at line 89 of file GUIController.java.

16.12.2.3 getDN()

```
String com.zegline.thubot.core.controller.GUIController.getDN ( )
```

End point to serve the database display page.

Returns

String representing name of the view for database display page.

Definition at line 67 of file GUIController.java.

16.12.2.4 getDN1()

```
String com.zegline.thubot.core.controller.GUIController.getDN1 ( )
```

End point to serve the static database display page.

Returns

String representing name of the view for static database display page.

Definition at line 77 of file GUIController.java.

16.12.2.5 getIndex()

```
String com.zegline.thubot.core.controller.GUIController.getIndex (
            Model model )
```

Serves the main index page and populates it with actuator information.

Parameters

model	The model in which to place attributes for the view.

Returns

String representing name of the view for the main index page.

Definition at line 55 of file GUIController.java.

16.12.2.6 login()

```
String com.zegline.thubot.core.controller.GUIController.login (
            Model model )
```

Serves the login view.

Parameters

model	The model in which to place attributes for the view.

Returns

Name of the view for the login page.

Definition at line 103 of file GUIController.java.

16.12.2.7 registerUser()

```
String com.zegline.thubot.core.controller.GUIController.registerUser (
            Model model,
            @RequestParam Map< String, String > body )
```

Handles the user registration process, including error handling and successful registration messaging.

Parameters

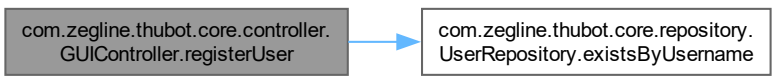
model	The model in which to place attributes for the view.
body	The form data from the registration form.

Returns

Name of the view for the registration page (to be refreshed with new messages or errors).

Definition at line 140 of file GUIController.java.

Here is the call graph for this function:



16.12.2.8 showRegisterForm()

```
String com.zegline.thubot.core.controller.GUIController.showRegisterForm (
            Model model )
```

Serves the registration form view.

Parameters

model	The model in which to place attributes for the view.

Returns

Name of the view for the registration form.

Definition at line 116 of file GUIController.java.

16.12.3 Member Data Documentation

16.12.3.1 infoEndpoint

```
InfoEndpoint com.zegline.thubot.core.controller.GUIController.infoEndpoint  [private]
```

Definition at line 46 of file GUIController.java.

16.12.3.2 passwordEncoder

```
PasswordEncoder com.zegline.thubot.core.controller.GUIController.passwordEncoder  [private]
```

Definition at line 43 of file GUIController.java.

16.12.3.3 ur

```
UserRepository com.zegline.thubot.core.controller.GUIController.ur  [private]
```

Definition at line 40 of file GUIController.java.

The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/controller/GUIController.java

16.13 com.zegline.thubot.core.service.openai.OpenAIService Class Reference

Service class for OpenAI API interaction.

Public Member Functions

- List< String > getQuestionMatch (String input_question, List< String > list_nodes)

Private Attributes

- String openaiApiKey

16.13.1 Detailed Description

Service class for OpenAI API interaction.

Responsible for interacting with the OpenAI API. Constructs HTTP POST requests to the API, sends the requests, and processes the API response.

Definition at line 32 of file OpenAIService.java.

16.13.2 Member Function Documentation

16.13.2.1 getQuestionMatch()

```
List< String > com.zegline.thubot.core.service.openai.OpenAIService.getQuestionMatch (
            String input_question,
            List< String > list_nodes )
```

Takes a user provided question and a list of chat node questions then gets the question that best matches the user's question from the OpenAI API.

Parameters

input_question	The user's question.
list_nodes	A list of possible response questions.

Returns

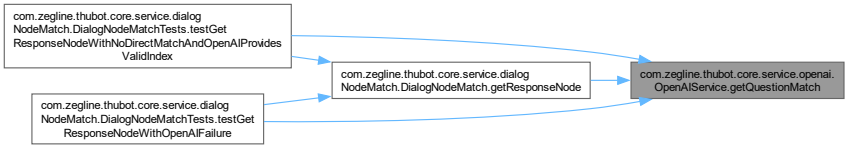
A list containing the best matching question.

Exceptions

Exception	Throws an exception if there was a problem with the API request.

Definition at line 47 of file OpenAIService.java.

Here is the caller graph for this function:



16.13.3 Member Data Documentation

16.13.3.1 openaiApiKey

```
String com.zegline.thubot.core.service.openai.OpenAIService.openaiApiKey  [private]
```

Definition at line 35 of file OpenAIService.java.

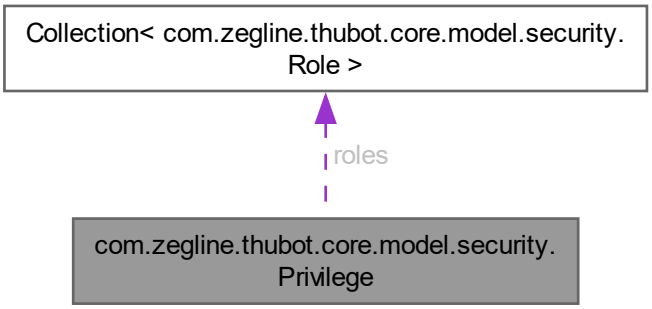
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/service/openai/OpenAIService.java

16.14 com.zegline.thubot.core.model.security.Privilege Class Reference

Entity to represent a specific privilege.

Collaboration diagram for com.zegline.thubot.core.model.security.Privilege:



Public Member Functions

- Privilege (String name)
- Privilege ()

Private Attributes

- Long id
- String name
- Collection< Role > roles

16.14.1 Detailed Description

Entity to represent a specific privilege.

The Privilege class represents a specific grantable access or action privilege held by a User. These privileges are grouped into Roles and can be used for role-based access control.

Definition at line 23 of file Privilege.java.

16.14.2 Constructor & Destructor Documentation

16.14.2.1 Privilege() [1/2]

```
com.zegline.thubot.core.model.security.Privilege.Privilege (
            String name )
```

Constructor method for a Privilege with a named parameter.

Parameters

name	The name for the privilege.

Definition at line 49 of file Privilege.java.

16.14.2.2 Privilege() [2/2]

```
com.zegline.thubot.core.model.security.Privilege.Privilege ( )
```

Default constructor for a Privilege.

Definition at line 56 of file Privilege.java.

16.14.3 Member Data Documentation

16.14.3.1 id

```
Long com.zegline.thubot.core.model.security.Privilege.id  [private]
```

Unique ID for the privilege.

Definition at line 30 of file Privilege.java.

16.14.3.2 name

```
String com.zegline.thubot.core.model.security.Privilege.name  [private]
```

The name of the privilege.

Definition at line 36 of file Privilege.java.

16.14.3.3 roles

```
Collection<Role> com.zegline.thubot.core.model.security.Privilege.roles  [private]
```

Collection of roles that have this privilege.

Definition at line 42 of file Privilege.java.

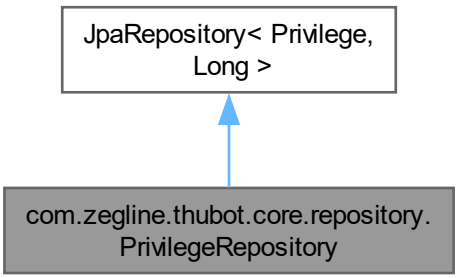
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/model/security/Privilege.java

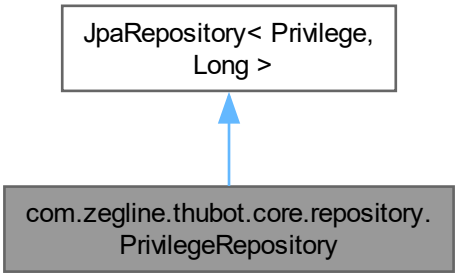
16.15 com.zegline.thubot.core.repository.PrivilegeRepository Interface Reference

Interface providing CRUD operations and custom queries for Privilege entities.

Inheritance diagram for com.zegline.thubot.core.repository.PrivilegeRepository:



Collaboration diagram for com.zegline.thubot.core.repository.PrivilegeRepository:



Public Member Functions

- Privilege findByName (String name)

16.15.1 Detailed Description

Interface providing CRUD operations and custom queries for Privilege entities.

Definition at line 17 of file PrivilegeRepository.java.

16.15.2 Member Function Documentation

16.15.2.1 findByName()

```
Privilege com.zegline.thubot.core.repository.PrivilegeRepository.findByName (
            String name )
```

Finds a Privilege by its name.

Parameters

name	The name of the Privilege to find.

Returns

The Privilege instance with the specified name.

Here is the caller graph for this function:

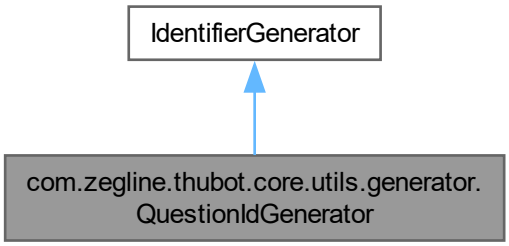
The documentation for this interface was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/repository/PrivilegeRepository.java

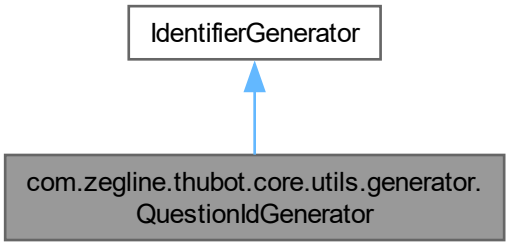
16.16 com.zegline.thubot.core.utils.generator.QuestionIdGenerator Class Reference

Custom ID generator for Hibernate entities.

Inheritance diagram for com.zegline.thubot.core.utils.generator.QuestionIdGenerator:



Collaboration diagram for com.zegline.thubot.core.utils.generator.QuestionIdGenerator:



Public Member Functions

- Serializable generate (SharedSessionContractImplementor session, Object object)
- void configure (Type type, Properties params, ServiceRegistry serviceRegistry) throws MappingException
- void setPrefix (String p)

Private Member Functions

- boolean isIdExists (SharedSessionContractImplementor session, String generatedId)

Private Attributes

- String prefix

16.16.1 Detailed Description

Custom ID generator for Hibernate entities.

QuestionIdGenerator extends Hibernate's IdentifierGenerator to create unique IDs with a custom prefix followed by a random number. It is used to generate IDs that do not collide with existing entities

Definition at line 29 of file QuestionIdGenerator.java.

16.16.2 Member Function Documentation

16.16.2.1 configure()

```
void com.zegline.thubot.core.utils.generator.QuestionIdGenerator.configure (
            Type type,
            Properties params,
            ServiceRegistry serviceRegistry ) throws MappingException
```

Configures the generator with necessary parameters

This method configures the ID generator, setting the prefix that will be used when generating new IDs

Parameters

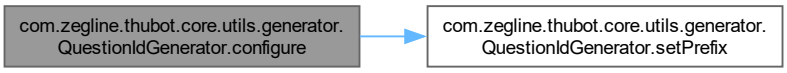
type	The type of the entity for which the ID will be generated
params	The parameters required for configuration, including the prefix
serviceRegistry	The service registry

Exceptions

MappingException	If there is an issue with the configuration such as missing or invalid parameters

Definition at line 81 of file QuestionIdGenerator.java.

Here is the call graph for this function:



16.16.2.2 generate()

```
Serializable com.zegline.thubot.core.utils.generator.QuestionIdGenerator.generate (
            SharedSessionContractImplementor session,
            Object object )
```

Generates a unique identifier for an entity

This method attempts to create a unique ID by appending a randomly generated number to a predefined prefix. It repeats the process until it finds an ID that does not already exist in the database

Parameters

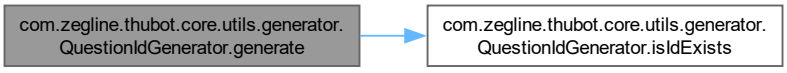
session	The Hibernate session currently associated with the transaction
object	The entity for which the ID is being generated

Returns

A unique identifier as a Serializable object

Definition at line 44 of file QuestionIdGenerator.java.

Here is the call graph for this function:



16.16.2.3 isIdExists()

```
boolean com.zegline.thubot.core.utils.generator.QuestionIdGenerator.isIdExists (
            SharedSessionContractImplementor session,
            String generatedId )  [private]
```

Checks the existence of a generated ID in the database

This method queries the database to see if an ID already exists to ensure uniqueness of the generated IDs

Parameters

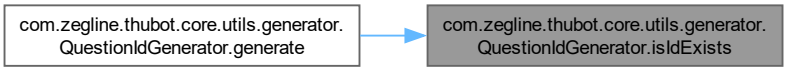
session	The Hibernate session currently associated with the transaction
generated↩ Id	The ID to be checked

Returns

True if the ID exists, false otherwise

Definition at line 63 of file QuestionIdGenerator.java.

Here is the caller graph for this function:



16.16.2.4 setPrefix()

```
void com.zegline.thubot.core.utils.generator.QuestionIdGenerator.setPrefix (
            String p )
```

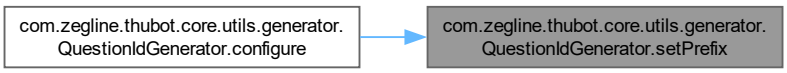
Sets the prefix to be used when generating new IDs

Parameters

p	The prefix to be used in ID generation

Definition at line 90 of file QuestionIdGenerator.java.

Here is the caller graph for this function:



16.16.3 Member Data Documentation

16.16.3.1 prefix

```
String com.zegline.thubot.core.utils.generator.QuestionIdGenerator.prefix  [private]
```

Definition at line 31 of file QuestionIdGenerator.java.

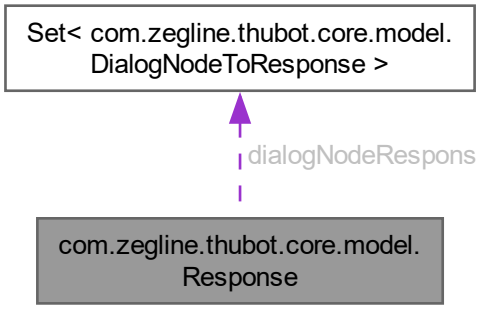
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/utils/generator/QuestionIdGenerator.java

16.17 com.zegline.thubot.core.model.Response Class Reference

Entity representing a response in the system.

Collaboration diagram for com.zegline.thubot.core.model.Response:



Public Member Functions

- Response ()
- Response (String r)
- String getResponseText ()
- void setResponseText (String rt)

Package Attributes

- Set< DialogNodeToResponse > dialogNodeRespons

Private Attributes

- long id
- String response_text

16.17.1 Detailed Description

Entity representing a response in the system.

The Response entity contains the response text that can be associated with multiple dialog nodes. The associations are maintained through a set of DialogNodeToResponse entities which map each response to various dialog nodes.

Definition at line 27 of file Response.java.

16.17.2 Constructor & Destructor Documentation

16.17.2.1 Response() [1/2]

```
com.zegline.thubot.core.model.Response.Response ( )
```

Default Constructor for Response

Definition at line 42 of file Response.java.

16.17.2.2 Response() [2/2]

```
com.zegline.thubot.core.model.Response.Response (
            String r )
```

Constructor for Response with response text initialization.

Parameters

r	The response text to initialize.

Definition at line 49 of file Response.java.

16.17.3 Member Function Documentation

16.17.3.1 getResponseText()

```
String com.zegline.thubot.core.model.Response.getResponseText ( )
```

Returns the response text.

Returns

The text of the response.

Definition at line 58 of file Response.java.

16.17.3.2 setResponseText()

```
void com.zegline.thubot.core.model.Response.setResponseText (
            String rt )
```

Updates the response text.

Parameters

rt	The text to set as the response.

Definition at line 67 of file Response.java.

16.17.4 Member Data Documentation

16.17.4.1 dialogNodeRespons

```
Set<DialogNodeToResponse> com.zegline.thubot.core.model.Response.dialogNodeRespons  [package]
```

Definition at line 37 of file Response.java.

16.17.4.2 id

```
long com.zegline.thubot.core.model.Response.id  [private]
```

Definition at line 31 of file Response.java.

16.17.4.3 response_text

```
String com.zegline.thubot.core.model.Response.response_text  [private]
```

Definition at line 34 of file Response.java.

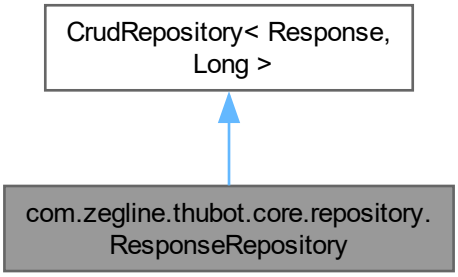
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/model/Response.java

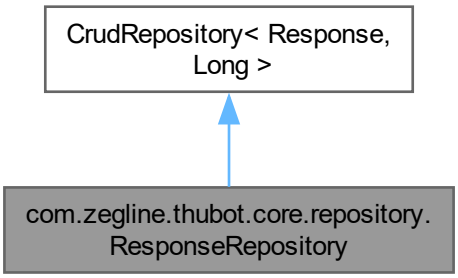
16.18 com.zegline.thubot.core.repository.ResponseRepository Interface Reference

Repository interface providing CRUD operations for Response entities.

Inheritance diagram for com.zegline.thubot.core.repository.ResponseRepository:



Collaboration diagram for com.zegline.thubot.core.repository.ResponseRepository:



16.18.1 Detailed Description

Repository interface providing CRUD operations for Response entities.

By extending CrudRepository, this interface automatically inherits several methods for working with Response data, such as saving, deleting, and finding Response entities.

Definition at line 22 of file ResponseRepository.java.

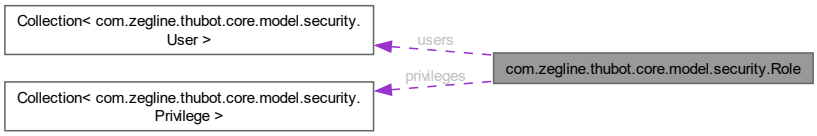
The documentation for this interface was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/repository/ResponseRepository.java

16.19 com.zegline.thubot.core.model.security.Role Class Reference

Entity to represent a specific role.

Collaboration diagram for com.zegline.thubot.core.model.security.Role:



Public Member Functions

- Role (String name)
- Role ()
- void setPrivileges (Collection< Privilege > privileges)

Private Attributes

- Long id
- String name
- Collection< User > users
- Collection< Privilege > privileges

16.19.1 Detailed Description

Entity to represent a specific role.

The Role class represents a specific role that a User can have in the system. Each role is essentially a collection of Privileges, and thus can be used for role-based access control.

Definition at line 25 of file Role.java.

16.19.2 Constructor & Destructor Documentation

16.19.2.1 Role() [1/2]

```
com.zegline.thubot.core.model.security.Role.Role (
            String name )
```

Constructor method for a Role with a named parameter.

Parameters

name	The name for the role.

Definition at line 68 of file Role.java.

16.19.2.2 Role() [2/2]

```
com.zegline.thubot.core.model.security.Role.Role ( )
```

Default constructor for a Role.

Definition at line 75 of file Role.java.

16.19.3 Member Function Documentation

16.19.3.1 setPrivileges()

```
void com.zegline.thubot.core.model.security.Role.setPrivileges (
            Collection< Privilege > privileges )
```

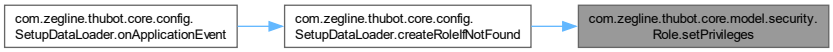
Method to set privileges to the role

Parameters

privileges	Collection of Privilege entities

Definition at line 83 of file Role.java.

Here is the caller graph for this function:



16.19.4 Member Data Documentation

16.19.4.1 id

```
Long com.zegline.thubot.core.model.security.Role.id  [private]
```

Unique ID for the role.

Definition at line 32 of file Role.java.

16.19.4.2 name

```
String com.zegline.thubot.core.model.security.Role.name  [private]
```

The name of the role.

Definition at line 38 of file Role.java.

16.19.4.3 privileges

```
Collection<Privilege> com.zegline.thubot.core.model.security.Role.privileges  [private]
```

Collection of privileges that this role holds.

Definition at line 61 of file Role.java.

16.19.4.4 users

```
Collection<User> com.zegline.thubot.core.model.security.Role.users  [private]
```

Collection of users who have this role.

Definition at line 44 of file Role.java.

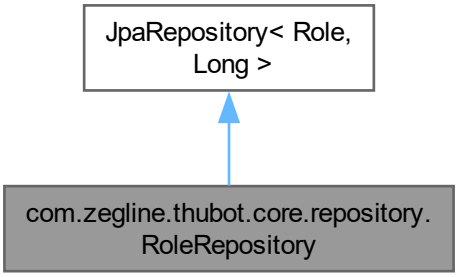
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/model/security/Role.java

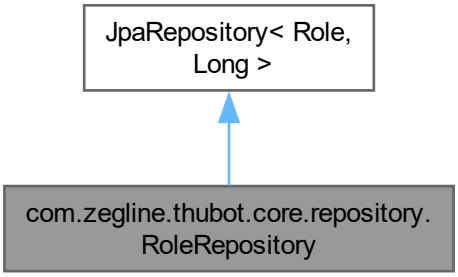
16.20 com.zegline.thubot.core.repository.RoleRepository Interface Reference

Repository interface providing CRUD operations and a method to find Role by name.

Inheritance diagram for com.zegline.thubot.core.repository.RoleRepository:



Collaboration diagram for com.zegline.thubot.core.repository.RoleRepository:



Public Member Functions

- Role findByName (String name)

16.20.1 Detailed Description

Repository interface providing CRUD operations and a method to find Role by name.

By extending JpaRepository, this interface automatically inherits several methods for working with Role data, such as saving, deleting, finding, and paging through Role entities. Additionally, this interface provides a function to find a Role entity by its name.

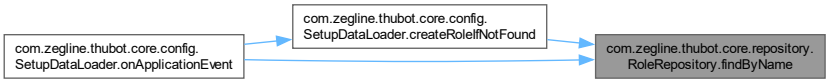
Definition at line 21 of file RoleRepository.java.

16.20.2 Member Function Documentation

16.20.2.1 findByName()

```
Role com.zegline.thubot.core.repository.RoleRepository.findByName (
            String name )
```

Here is the caller graph for this function:



The documentation for this interface was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/repository/RoleRepository.java

16.21 com.zegline.thubot.core.config.SecurityConfig Class Reference

Configures the security settings including security filters, user details, and password encoding.

Package Functions

- SecurityFilterChain filterChain (HttpSecurity http) throws Exception
- PasswordEncoder passwordEncoder ()

16.21.1 Detailed Description

Configures the security settings including security filters, user details, and password encoding.

This class defines the beans for setting up the security filters, defines users with their roles and authorities, and encodes their passwords by configuring the PasswordEncoder. The user details are managed in-memory for simplicity.

Definition at line 28 of file SecurityConfig.java.

16.21.2 Member Function Documentation

16.21.2.1 filterChain()

```
SecurityFilterChain com.zegline.thubot.core.config.SecurityConfig.filterChain (
            HttpSecurity http ) throws Exception  [package]
```

Configures the security filter chain to set up CORS, CSRF, and endpoint access rules

Parameters

http	HttpSecurity to configure the request authorizations and access rules

Returns

The configured SecurityFilterChain

Exceptions

Exception	if an error occurs during configuration

Definition at line 38 of file SecurityConfig.java.

16.21.2.2 passwordEncoder()

```
PasswordEncoder com.zegline.thubot.core.config.SecurityConfig.passwordEncoder ( )  [package]
```

Configures the password encoder to be used for encoding and matching passwords

Returns

A PasswordEncoder that uses BCrypt hashing for securing passwords

Definition at line 68 of file SecurityConfig.java.

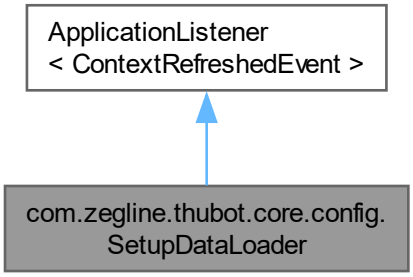
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/config/SecurityConfig.java

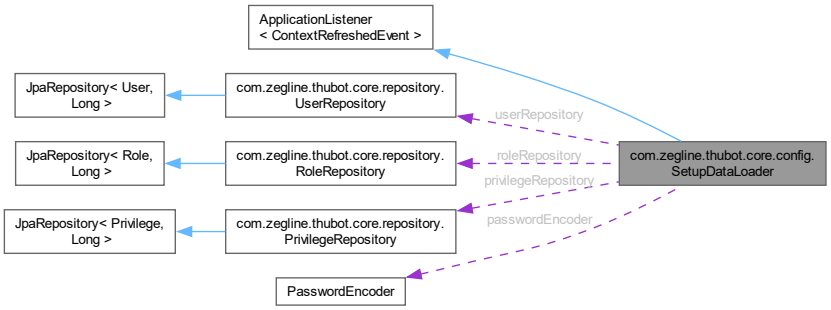
16.22 com.zegline.thubot.core.config.SetupDataLoader Class Reference

This class initiates some database content on application startup.

Inheritance diagram for com.zegline.thubot.core.config.SetupDataLoader:



Collaboration diagram for com.zegline.thubot.core.config.SetupDataLoader:



Public Member Functions

- void onApplicationEvent (ContextRefreshedEvent event)
- Privilege createPrivilegeIfNotFound (String name)
- Role createRoleIfNotFound (String name, Collection< Privilege > privileges)

Package Attributes

- boolean alreadySetup = false

Private Attributes

- UserRepository userRepository
- RoleRepository roleRepository
- PrivilegeRepository privilegeRepository
- PasswordEncoder passwordEncoder

16.22.1 Detailed Description

This class initiates some database content on application startup.

It implements ApplicationListener<ContextRefreshedEvent> to run setup logic after the Spring context is initialized. Privileges, roles and a test user are created if they don't exist. This class avoids repeating setup after the first run.

Definition at line 36 of file SetupDataLoader.java.

16.22.2 Member Function Documentation

16.22.2.1 createPrivilegeIfNotFound()

```
Privilege com.zegline.thubot.core.config.SetupDataLoader.createPrivilegeIfNotFound (
            String name )
```

Creates a new privilege in the database if it does not exist.

Parameters

name	the name of the privilege

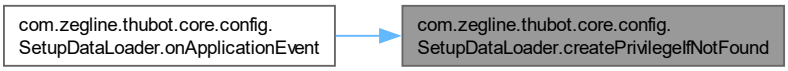
Returns

the existing or newly created privilege

Definition at line 95 of file SetupDataLoader.java.

Here is the call graph for this function:

Here is the caller graph for this function:



16.22.2.2 createRoleIfNotFound()

```
Role com.zegline.thubot.core.config.SetupDataLoader.createRoleIfNotFound (
            String name,
            Collection< Privilege > privileges )
```

Creates a new role in the database if it does not exist.

Parameters

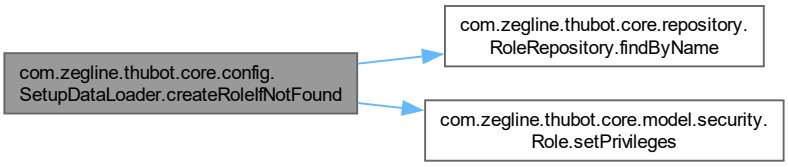
name	the name of the role
privileges	the privileges associated with the role

Returns

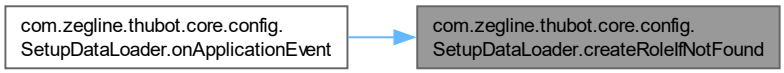
the existing or newly created role

Definition at line 113 of file SetupDataLoader.java.

Here is the call graph for this function:



Here is the caller graph for this function:



16.22.2.3 onApplicationEvent()

```
void com.zegline.thubot.core.config.SetupDataLoader.onApplicationEvent (
            ContextRefreshedEvent event )
```

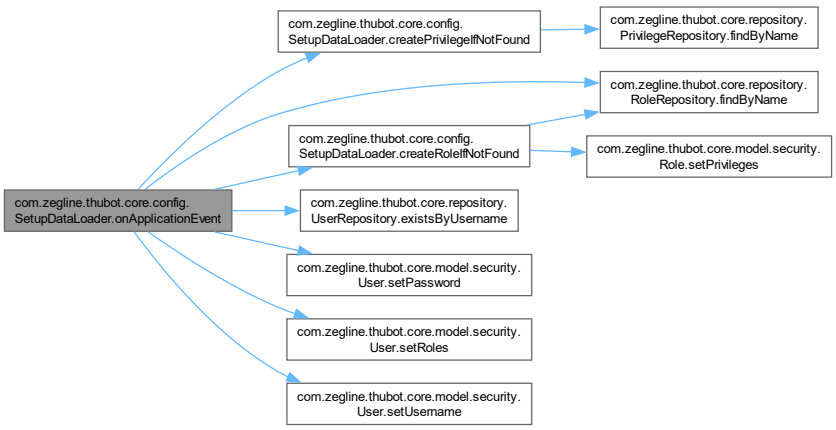
This event is executed when the application context is refreshed. It sets up initial data in the database if not already setup.

Parameters

event	triggered when the application context is started or refreshed

Definition at line 60 of file SetupDataLoader.java.

Here is the call graph for this function:



16.22.3 Member Data Documentation

16.22.3.1 alreadySetup

```
boolean com.zegline.thubot.core.config.SetupDataLoader.alreadySetup = false  [package]
```

Definition at line 38 of file SetupDataLoader.java.

16.22.3.2 passwordEncoder

```
PasswordEncoder com.zegline.thubot.core.config.SetupDataLoader.passwordEncoder  [private]
```

Definition at line 50 of file SetupDataLoader.java.

16.22.3.3 privilegeRepository

```
PrivilegeRepository com.zegline.thubot.core.config.SetupDataLoader.privilegeRepository  [private]
```

Definition at line 47 of file SetupDataLoader.java.

16.22.3.4 roleRepository

```
RoleRepository com.zegline.thubot.core.config.SetupDataLoader.roleRepository  [private]
```

Definition at line 44 of file SetupDataLoader.java.

16.22.3.5 userRepository

```
UserRepository com.zegline.thubot.core.config.SetupDataLoader.userRepository  [private]
```

Definition at line 41 of file SetupDataLoader.java.

The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/config/SetupDataLoader.java

16.23 com.zegline.thubot.ThuBotApplication Class Reference

Main application class for the ThuBot chatbot.

Static Public Member Functions

- static void main (String[ ] args)

16.23.1 Detailed Description

Main application class for the ThuBot chatbot.

This is the main class that bootstraps the Spring Boot application. It contains the main method which is the entry point of the Java application

Definition at line 22 of file ThuBotApplication.java.

16.23.2 Member Function Documentation

16.23.2.1 main()

```
static void com.zegline.thubot.ThuBotApplication.main (
            String[] args )  [static]
```

The main method that starts up the Spring Boot application.

Parameters

args	A string array containing command-line arguments that were passed to this application. The Spring Boot SpringApplication.run() method is called inside the main method, which initializes the framework and starts up the embedded web server (provided by Spring Boot) allowing the application to serve HTTP requests.

Definition at line 31 of file ThuBotApplication.java.

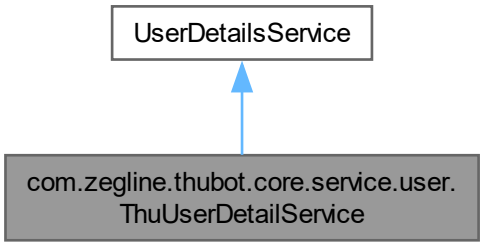
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/ThuBotApplication.java

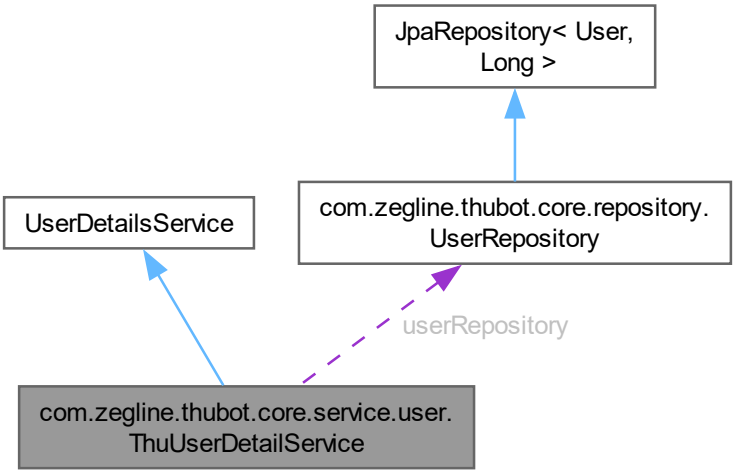
16.24 com.zegline.thubot.core.service.user.ThuUserDetailService Class Reference

Service class for loading user-specific data needed for authentication.

Inheritance diagram for com.zegline.thubot.core.service.user.ThuUserDetailService:



Collaboration diagram for com.zegline.thubot.core.service.user.ThuUserDetailService:



Public Member Functions

- UserDetails loadUserByUsername (String username) throws UsernameNotFoundException

Private Attributes

- UserRepository userRepository

16.24.1 Detailed Description

Service class for loading user-specific data needed for authentication.

Implements UserDetailsService from the Spring Security framework to provide authentication services. It uses a UserRepository to retrieve User entities from the database.

Definition at line 27 of file ThuUserDetailService.java.

16.24.2 Member Function Documentation

16.24.2.1 loadUserByUsername()

```
UserDetails com.zegline.thubot.core.service.user.ThuUserDetailService.loadUserByUsername (
            String username ) throws UsernameNotFoundException
```

Loads the user entity by its username.

Parameters

username	The username identifying the user.

Returns

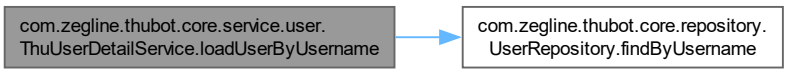
UserDetails object that Spring Security can use for authentication and validation.

Exceptions

UsernameNotFoundException	if the user entity was not found.

Definition at line 40 of file ThuUserDetailService.java.

Here is the call graph for this function:



16.24.3 Member Data Documentation

16.24.3.1 userRepository

```
UserRepository com.zegline.thubot.core.service.user.ThuUserDetailService.userRepository  [private]
```

Definition at line 30 of file ThuUserDetailService.java.

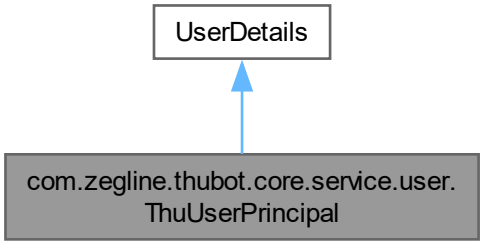
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/service/user/ThuUserDetailService.java

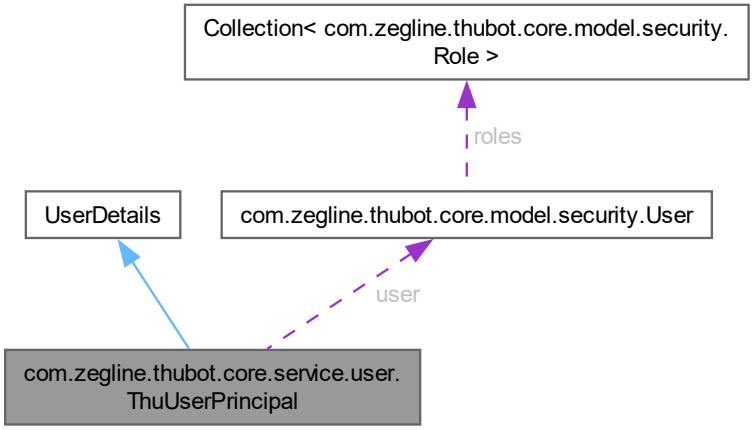
16.25 com.zegline.thubot.core.service.user.ThuUserPrincipal Class Reference

Implements UserDetails interface for User authentication.

Inheritance diagram for com.zegline.thubot.core.service.user.ThuUserPrincipal:



Collaboration diagram for com.zegline.thubot.core.service.user.ThuUserPrincipal:



Public Member Functions

- ThuUserPrincipal (User user)
- Collection<? extends GrantedAuthority > getAuthorities ()
- String getPassword ()
- String getUsername ()
- boolean isAccountNonExpired ()
- boolean isAccountNonLocked ()
- boolean isCredentialsNonExpired ()
- boolean isEnabled ()

Private Member Functions

- List< String > getPrivileges (Collection< Role > roles)
- List< GrantedAuthority > getGrantedAuthorities (List< String > privileges)

Private Attributes

- User user

16.25.1 Detailed Description

Implements UserDetails interface for User authentication.

Wraps around a User entity and provides details needed for authentication.

Definition at line 28 of file ThuUserPrincipal.java.

16.25.2 Constructor & Destructor Documentation

16.25.2.1 ThuUserPrincipal()

```
com.zegline.thubot.core.service.user.ThuUserPrincipal.ThuUserPrincipal (
            User user )
```

Constructor that accepts a User entity.

Parameters

user	The User entity to be used for authentication.

Definition at line 37 of file ThuUserPrincipal.java.

16.25.3 Member Function Documentation

16.25.3.1 getAuthorities()

```
Collection<? extends GrantedAuthority > com.zegline.thubot.core.service.user.ThuUserPrincipal.↩
getAuthorities ( )
```

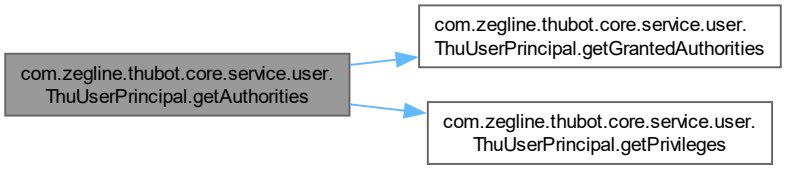
Returns the authorities granted to the user. Need not be overridden unless for more complex role and privilege setups.

Returns

A collection of GrantedAuthority which represents the user's authorities.

Definition at line 48 of file ThuUserPrincipal.java.

Here is the call graph for this function:



16.25.3.2 getGrantedAuthorities()

```
List< GrantedAuthority > com.zegline.thubot.core.service.user.ThuUserPrincipal.getGranted↩
Authorities (
            List< String > privileges )  [private]
```

Creates a list of GrantedAuthorities from the list of privileges.

Parameters

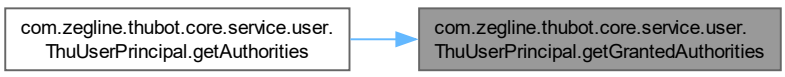
privileges	A list of privileges as Strings.

Returns

A list of GrantedAuthority created from the privileges.

Definition at line 78 of file ThuUserPrincipal.java.

Here is the caller graph for this function:



16.25.3.3 getPassword()

```
String com.zegline.thubot.core.service.user.ThuUserPrincipal.getPassword ( )
```

Returns the password associated with the user.

Returns

A string representing the user's password.

Definition at line 92 of file ThuUserPrincipal.java.

16.25.3.4 getPrivileges()

```
List< String > com.zegline.thubot.core.service.user.ThuUserPrincipal.getPrivileges (
            Collection< Role > roles )  [private]
```

Obtains all the privileges from the collection of roles associated with the user.

Parameters

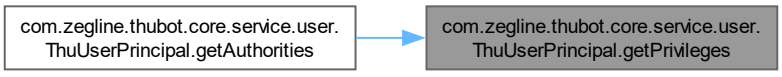
roles	A collection of Role entities linked with the user.

Returns

A list of privileges as Strings.

Definition at line 58 of file ThuUserPrincipal.java.

Here is the caller graph for this function:



16.25.3.5 getUsername()

```
String com.zegline.thubot.core.service.user.ThuUserPrincipal.getUsername ( )
```

Returns the username of the user.

Returns

A string representing the user's username.

Definition at line 102 of file ThuUserPrincipal.java.

16.25.3.6 isAccountNonExpired()

```
boolean com.zegline.thubot.core.service.user.ThuUserPrincipal.isAccountNonExpired ( )
```

Checks if the user's account has not expired.

Returns

always true in this implementation.

Definition at line 112 of file ThuUserPrincipal.java.

16.25.3.7 isAccountNonLocked()

```
boolean com.zegline.thubot.core.service.user.ThuUserPrincipal.isAccountNonLocked ( )
```

Checks if the user's account is not locked.

Returns

always true in this implementation.

Definition at line 122 of file ThuUserPrincipal.java.

16.25.3.8 isCredentialsNonExpired()

```
boolean com.zegline.thubot.core.service.user.ThuUserPrincipal.isCredentialsNonExpired ( )
```

Checks if the user's credentials (password) has not expired.

Returns

always true in this implementation.

Definition at line 132 of file ThuUserPrincipal.java.

16.25.3.9 isEnabled()

```
boolean com.zegline.thubot.core.service.user.ThuUserPrincipal.isEnabled ( )
```

Checks if the user's account is enabled.

Returns

always true in this implementation.

Definition at line 142 of file ThuUserPrincipal.java.

16.25.4 Member Data Documentation

16.25.4.1 user

```
User com.zegline.thubot.core.service.user.ThuUserPrincipal.user  [private]
```

Definition at line 30 of file ThuUserPrincipal.java.

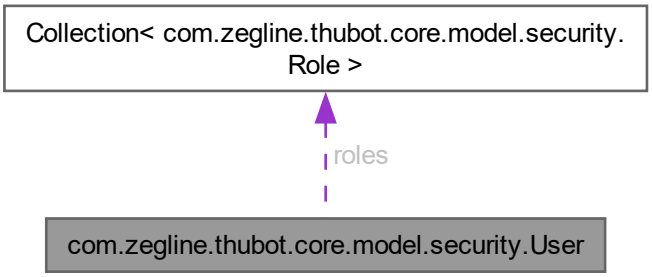
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/service/user/ThuUserPrincipal.java

16.26 com.zegline.thubot.core.model.security.User Class Reference

Entity representing a user.

Collaboration diagram for com.zegline.thubot.core.model.security.User:



Public Member Functions

- void setUsername (String username)
- void setPassword (String password)
- void setRoles (Collection< Role > roles)

Private Attributes

- Long id
- String username
- String password
- Collection< Role > roles

16.26.1 Detailed Description

Entity representing a user.

The User class represents a specific user of the system. The User's information includes a unique id, username, and password, along with a set of roles that dictate access control for the user.

Definition at line 26 of file User.java.

16.26.2 Member Function Documentation

16.26.2.1 setPassword()

```
void com.zegline.thubot.core.model.security.User.setPassword (
            String password )
```

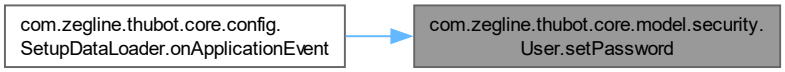
Method to set the User's password.

Parameters

password	The User's password.

Definition at line 79 of file User.java.

Here is the caller graph for this function:



16.26.2.2 setRoles()

```
void com.zegline.thubot.core.model.security.User.setRoles (
            Collection< Role > roles )
```

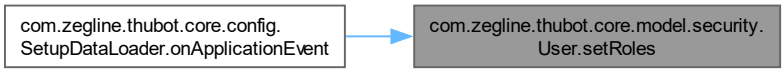
Method to set the roles of the User.

Parameters

roles	The roles to be associated with the User.

Definition at line 88 of file User.java.

Here is the caller graph for this function:



16.26.2.3 setUsername()

```
void com.zegline.thubot.core.model.security.User.setUsername (
            String username )
```

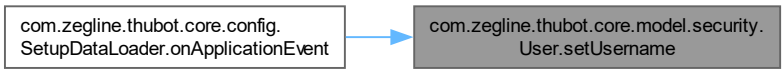
Method to set the User's username.

Parameters

username	The User's username.

Definition at line 70 of file User.java.

Here is the caller graph for this function:



16.26.3 Member Data Documentation

16.26.3.1 id

```
Long com.zegline.thubot.core.model.security.User.id  [private]
```

Unique identifier for the User.

Definition at line 33 of file User.java.

16.26.3.2 password

```
String com.zegline.thubot.core.model.security.User.password  [private]
```

The User's password.

Definition at line 46 of file User.java.

16.26.3.3 roles

```
Collection<Role> com.zegline.thubot.core.model.security.User.roles  [private]
```

Collection of the roles associated with the User.

Definition at line 63 of file User.java.

16.26.3.4 username

```
String com.zegline.thubot.core.model.security.User.username  [private]
```

The User's username. This field is unique and non-null.

Definition at line 40 of file User.java.

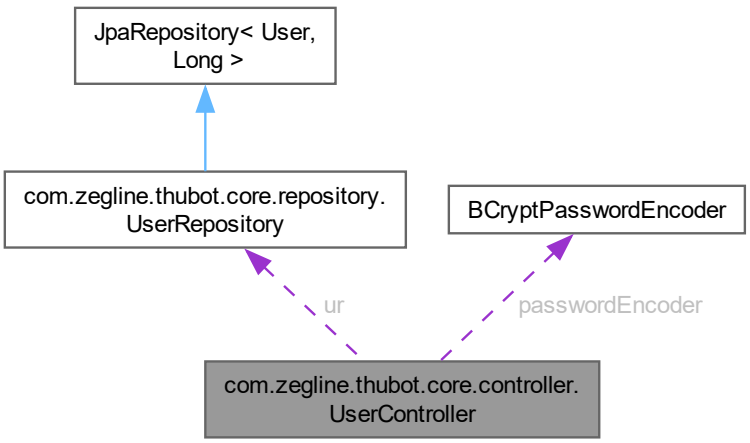
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/model/security/User.java

16.27 com.zegline.thubot.core.controller.UserController Class Reference

Controller class for User-related actions.

Collaboration diagram for com.zegline.thubot.core.controller.UserController:



Public Member Functions

- ResponseEntity< User > createUser (@RequestBody User user)

Package Attributes

- UserRepository ur

Private Attributes

- final BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder()

16.27.1 Detailed Description

Controller class for User-related actions.

Provides the REST endpoints for manipulating User entities. This class is only intended for testing purposes and should not be used in a staging or production environment.

Definition at line 30 of file UserController.java.

16.27.2 Member Function Documentation

16.27.2.1 createUser()

```
ResponseEntity< User > com.zegline.thubot.core.controller.UserController.createUser (
            @RequestBody User user )
```

Definition at line 42 of file UserController.java.

16.27.3 Member Data Documentation

16.27.3.1 passwordEncoder

```
final BCryptPasswordEncoder com.zegline.thubot.core.controller.UserController.passwordEncoder
= new BCryptPasswordEncoder()  [private]
```

Definition at line 35 of file UserController.java.

16.27.3.2 ur

```
UserRepository com.zegline.thubot.core.controller.UserController.ur  [package]
```

Definition at line 33 of file UserController.java.

The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/controller/UserController.java

16.28 com.zegline.thubot.core.controller.UserInputController Class Reference

Controller to manage user input related actions.

Collaboration diagram for com.zegline.thubot.core.controller.UserInputController:

Public Member Functions

- DialogNode inputAsk (@RequestParam String userInput)

Private Attributes

- String openaiApiKey
- DialogNodeMatch dialogNodeMatchService

16.28.1 Detailed Description

Controller to manage user input related actions.

Provides an API endpoint to receive user input and return a response. The response is either matched from dialog nodes or queried from OpenAI service.

Definition at line 29 of file UserInputController.java.

16.28.2 Member Function Documentation

16.28.2.1 inputAsk()

```
DialogNode com.zegline.thubot.core.controller.UserInputController.inputAsk (
            @RequestParam String userInput )
```

Endpoint to receive user input and retrieve a response node based on the provided input.

Parameters

userInput	The user input received as a request parameter.

Returns

The DialogNode instance that responds to the user input.

Definition at line 50 of file UserInputController.java.

16.28.3 Member Data Documentation

16.28.3.1 dialogNodeMatchService

```
DialogNodeMatch com.zegline.thubot.core.controller.UserInputController.dialogNodeMatchService
[private]
```

The service to match dialog nodes and interface with the OpenAI service.

Definition at line 41 of file UserInputController.java.

16.28.3.2 openaiApiKey

```
String com.zegline.thubot.core.controller.UserInputController.openaiApiKey  [private]
```

The OpenAI API key read from the application properties or the application.yml file.

Definition at line 35 of file UserInputController.java.

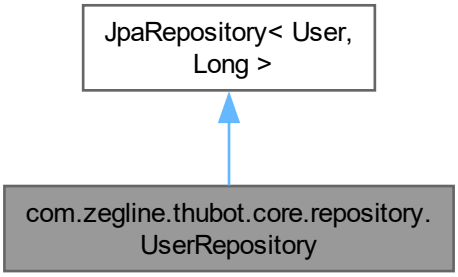
The documentation for this class was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/controller/UserInputController.java

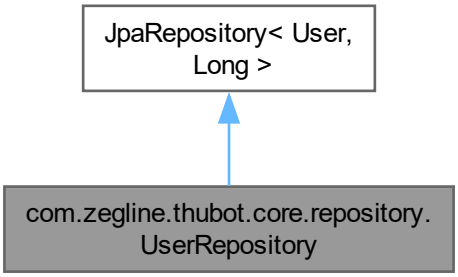
16.29 com.zegline.thubot.core.repository.UserRepository Interface Reference

Repository interface for User entities providing CRUD operations and additional methods.

Inheritance diagram for com.zegline.thubot.core.repository.UserRepository:



Collaboration diagram for com.zegline.thubot.core.repository.UserRepository:



Public Member Functions

- User findByUsername (String username)
- boolean existsByUsername (String username)

16.29.1 Detailed Description

Repository interface for User entities providing CRUD operations and additional methods.

By extending JpaRepository, this interface inherits several methods for working with User data such as saving, deleting, finding, and paging through User entities. Additionally, it provides a method to find a User by its username and to check if a User with a specified username exists.

Definition at line 18 of file UserRepository.java.

16.29.2 Member Function Documentation

16.29.2.1 existsByUsername()

```
boolean com.zegline.thubot.core.repository.UserRepository.existsByUsername (
            String username )
```

Checks if a User with a specified username exists.

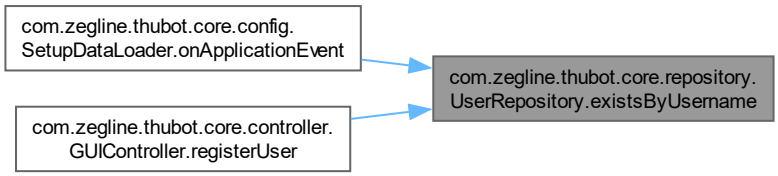
Parameters

username	- The username of the user.

Returns

true if a User with the supplied username exists, false otherwise.

Here is the caller graph for this function:



16.29.2.2 findByUsername()

```
User com.zegline.thubot.core.repository.UserRepository.findByUsername (
            String username )
```

Finds a User by its username.

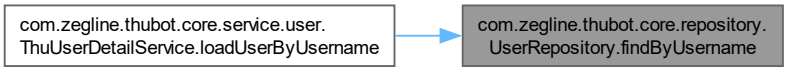
Parameters

username	- The username of the user.

Returns

The User with the supplied username.

Here is the caller graph for this function:



The documentation for this interface was generated from the following file:

- thuBOT/src/main/java/com/zegline/thubot/core/repository/UserRepository.java

Chapter 17

File Documentation

17.1 thuBOT/docs/architecture/api_routes.MD File Reference

17.2 thuBOT/docs/architecture/permissions_model.MD File Reference

17.3 thuBOT/docs/architecture/tree_vision.md File Reference

17.4 thuBOT/docs/contributing.MD File Reference

17.5 thuBOT/docs/increment1.md File Reference

17.6 thuBOT/docs/increment2.md File Reference

17.7 thuBOT/docs/requirements_specification/playground_openai.MD File Reference

17.8 thuBOT/docs/requirements_specification/team_roles.MD File Reference

17.9 thuBOT/docs/requirements_specification/technical_constraints.MD File Reference

17.10 thuBOT/README.md File Reference

17.11 thuBOT/src/main/java/com/zegline/thubot/core/config/SecurityConfig.java File Reference

Holds the configuration class for security purposes in the Spring application.

Classes

- class com.zegline.thubot.core.config.SecurityConfig

  Configures the security settings including security filters, user details, and password encoding.

Packages

- package com.zegline.thubot.core.config

17.11.1 Detailed Description

Holds the configuration class for security purposes in the Spring application.

This file contains the SecurityConfig class which provides the security configurations of the application. It configures the security filter chain, user details service, and password encoder.

Definition in file SecurityConfig.java.

17.12 SecurityConfig.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.config;
00009
00010 import org.springframework.context.annotation.Bean;
00011 import org.springframework.context.annotation.Configuration;
00012 import org.springframework.security.config.annotation.web.builders.HttpSecurity;
00013 import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
00014 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
00015 import org.springframework.security.crypto.password.PasswordEncoder;
00016 import org.springframework.security.web.SecurityFilterChain;
00017 import org.springframework.security.web.header.writers.StaticHeadersWriter;
00018
00026 @Configuration
00027 @EnableWebSecurity
00028 public class SecurityConfig {
00029
00037     @Bean
00038     SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
00039         http
00040             .cors(cors -> cors.disable())
00041             .csrf(csrf -> csrf.disable())
00042             .headers(headers -> headers.addHeaderWriter(new StaticHeadersWriter("X-Frame-Options",
      "SAMEORIGIN")))
00043             .authorizeHttpRequests(authz -> {
00044                 authz.requestMatchers("/database/display/**",
      "/database/display/static/**").permitAll();
00045                 authz.requestMatchers("/database/**").hasRole("SYS");
00046                 authz.requestMatchers("/api/input/**").permitAll();
00047                 authz.requestMatchers("/api/dialognode/get").permitAll();
00048                 authz.requestMatchers("/api/**").authenticated();
00049                 authz.anyRequest().permitAll();
00050             })
00051             .formLogin(form -> form
00052                 .loginPage("/login")
00053                 .permitAll())
00054             .exceptionHandling(exceptionHandling -> exceptionHandling
00055                 .accessDeniedPage("/access-denied")
00056             );
00057         return http.build();
00058
00059     }
00060
00061
00067     @Bean
00068     PasswordEncoder passwordEncoder() {
00069         return new BCryptPasswordEncoder();
00070     }
00071
00072
00073 }
```

17.13 thuBOT/src/main/java/com/zegline/thubot/core/config/SetupDataLoader.java File Reference

Data Loader for setting up initial values in the database on application startup.

Classes

- class com.zegline.thubot.core.config.SetupDataLoader

  This class initiates some database content on application startup.

Packages

- package com.zegline.thubot.core.config

17.13.1 Detailed Description

Data Loader for setting up initial values in the database on application startup.

This class is responsible for populating the database with a set of initial values for user, privileges and roles upon application startup. This is only done once when the application is starting.

Definition in file SetupDataLoader.java.

17.14 SetupDataLoader.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.config;
00009
00010 import com.zegline.thubot.core.model.security.Privilege;
00011 import com.zegline.thubot.core.model.security.Role;
00012 import com.zegline.thubot.core.model.security.User;
00013 import com.zegline.thubot.core.repository.PrivilegeRepository;
00014 import com.zegline.thubot.core.repository.RoleRepository;
00015 import com.zegline.thubot.core.repository.UserRepository;
00016
00017 import org.springframework.beans.factory.annotation.Autowired;
00018 import org.springframework.context.ApplicationListener;
00019 import org.springframework.context.event.ContextRefreshedEvent;
00020 import org.springframework.security.crypto.password.PasswordEncoder;
00021 import org.springframework.stereotype.Component;
00022 import org.springframework.transaction.annotation.Transactional;
00023
00024 import java.util.Arrays;
00025 import java.util.Collection;
00026 import java.util.List;
00027
00035 @Component
00036 public class SetupDataLoader implements ApplicationListener<ContextRefreshedEvent> {
00037
00038     boolean alreadySetup = false;
00039
00040     @Autowired
00041     private UserRepository userRepository;
00042
00043     @Autowired
00044     private RoleRepository roleRepository;
00045
00046     @Autowired
00047     private PrivilegeRepository privilegeRepository;
00048
00049     @Autowired
```

```
00050     private PasswordEncoder passwordEncoder;
00051
00058     @Override
00059     @Transactional
00060     public void onApplicationEvent(ContextRefreshedEvent event) {
00061         if (alreadySetup)
00062             return;
00063
00064         if (userRepository.existsByUsername("test_sys")) {
00065             return;
00066         }
00067         Privilege readPrivilege
00068                 = createPrivilegeIfNotFound("READ_PRIVILEGE");
00069         Privilege writePrivilege
00070                 = createPrivilegeIfNotFound("WRITE_PRIVILEGE");
00071
00072         List<Privilege> sysPrivileges = Arrays.asList(
00073                 readPrivilege, writePrivilege);
00074         createRoleIfNotFound("ROLE_SYS", sysPrivileges);
00075         createRoleIfNotFound("ROLE_USER", Arrays.asList(readPrivilege));
00076
00077         Role sysRole = roleRepository.findByName("ROLE_SYS");
00078
00079         User user = new User();
00080         user.setUsername("test_sys");
00081
00082         user.setPassword(passwordEncoder.encode("xenopower"));
00083         user.setRoles(Arrays.asList(sysRole));
00084         userRepository.save(user);
00085         alreadySetup = true;
00086     }
00087
00094     @Transactional
00095     public Privilege createPrivilegeIfNotFound(String name) {
00096
00097         Privilege privilege = privilegeRepository.findByName(name);
00098         if (privilege == null) {
00099             privilege = new Privilege(name);
00100             privilegeRepository.save(privilege);
00101         }
00102         return privilege;
00103     }
00104
00112     @Transactional
00113     public Role createRoleIfNotFound(
00114             String name, Collection<Privilege> privileges) {
00115
00116         Role role = roleRepository.findByName(name);
00117         if (role == null) {
00118             role = new Role(name);
00119             role.setPrivileges(privileges);
00120             roleRepository.save(role);
00121         }
00122         return role;
00123     }
00124 }
```

17.15 thuBOT/src/main/java/com/zegline/thubot/core/controller/DialogNodeController.java File Reference

Controller for handling requests related to DialogNodes. This controller provides REST endpoints for creating and retrieving DialogNodes which are components of a conversational interface.

Classes

- class com.zegline.thubot.core.controller.DialogNodeController

  Provides REST endpoints to manage DialogNodes.

Packages

- package com.zegline.thubot.core.controller

17.15.1 Detailed Description

Controller for handling requests related to DialogNodes. This controller provides REST endpoints for creating and retrieving DialogNodes which are components of a conversational interface.

Definition in file DialogNodeController.java.

17.16 DialogNodeController.java

Go to the documentation of this file.

```
00001
00007 package com.zegline.thubot.core.controller;
00008
00009 import java.util.*;
00010
00011 import org.springframework.beans.factory.annotation.Autowired;
00012 import org.springframework.http.HttpStatus;
00013 import org.springframework.web.bind.annotation.GetMapping;
00014 import org.springframework.web.bind.annotation.PostMapping;
00015 import org.springframework.web.bind.annotation.RequestBody;
00016 import org.springframework.web.bind.annotation.RequestMapping;
00017 import org.springframework.web.bind.annotation.RestController;
00018 import org.springframework.web.server.ResponseStatusException;
00019
00020 import com.zegline.thubot.core.model.DialogNode;
00021 import com.zegline.thubot.core.repository.DialogNodeRepository;
00022
00030 @RestController
00031 @RequestMapping("/api/dialognode")
00032 public class DialogNodeController {
00033
00034     @Autowired
00035     DialogNodeRepository dnr;
00036
00043     @PostMapping("/createChild")
00044     public DialogNode dialog_node_create(@RequestBody Map<String, String> body) {
00045         String dialogNodeText = body.get("dialogNodeText");
00046         String msgText = body.get("msgText");
00047         String parentNodeId = body.get("parentNodeId");
00048
00049         Optional<DialogNode> optionalParent = dnr.findById(parentNodeId);
00050         if (optionalParent.isPresent()) {
00051             DialogNode parent = optionalParent.get();
00052             DialogNode d = DialogNode.builder().dialogText(dialogNodeText).msgText(msgText).build();
00053             dnr.save(d);
00054             parent.addChild(d);
00055             dnr.save(parent);
00056             return parent;
00057         };
00058
00059         throw new ResponseStatusException(
00060                 HttpStatus.NOT_FOUND, "couldn't find parent"
00061         );
00062     }
00063
00071     @PostMapping("/modify")
00072     public DialogNode dialog_node_modify(@RequestBody Map<String, String> body) {
00073         String id = body.get("dialogNodeId");
00074         String newDialogNodeText = body.get("dialogNodeText");
00075         String newMsgText = body.get("msgText");
00076         String newParentNodeId = body.get("parentNodeId");
00077
00078         Optional<DialogNode> optionalNode = dnr.findById(id);
00079         if (optionalNode.isEmpty()) {
00080             throw new ResponseStatusException(
00081                     HttpStatus.NOT_FOUND, "couldn't find node"
00082             );
00083         }
00084
00085         DialogNode node = optionalNode.get();
00086         node.setMsgText(newMsgText);
00087         node.setDialogText(newDialogNodeText);
00088
00089         Optional<DialogNode> optionalNewParent = dnr.findById(newParentNodeId);
00090         if (optionalNewParent.isPresent()) {
00091             DialogNode newParent = optionalNewParent.get();
00092
00093             node.setParent(newParent);
```

```
00094         }
00095
00096         dnr.save(node);
00097         return node;
00098     }
00099
00108     @PostMapping("/delete")
00109     public DialogNode dialog_node_delete(@RequestBody Map<String, String> body){
00110         String id = body.get("dialogNodeId");
00111
00112         Optional<DialogNode> optionalNode = dnr.findById(id);
00113         if(optionalNode.isEmpty()) {
00114             throw new ResponseStatusException(
00115                     HttpStatus.NOT_FOUND, "couldn't find node"
00116             );
00117         }
00118
00119         DialogNode node = optionalNode.get();
00120         DialogNode nodeParent = node.getParent();
00121
00122         if (nodeParent == null) {
00123             throw new ResponseStatusException(
00124                     HttpStatus.FORBIDDEN, "Not allowed to delete root node"
00125             );
00126         }
00127
00128         if (!node.getChildren().isEmpty()) {
00129             throw new ResponseStatusException(
00130                     HttpStatus.FORBIDDEN, "cannot delete nodes with children"
00131             );
00132         }
00133
00134         nodeParent.getChildren().remove(node);
00135         dnr.delete(node);
00136         return nodeParent;
00137
00138     }
00139
00147     @GetMapping("/get")
00148     public Set<DialogNode> get(@RequestBody (required = false) Map<String, String> body) {
00149         Set<DialogNode> returned = new HashSet<>() ;
00150         if(body == null){
00151             List<DialogNode> nodes = dnr.findDialogNodesByParentIsNull();
00152
00153             for (DialogNode node : nodes) {
00154                 returned.add(node);
00155             }
00156
00157             return returned;
00158         }
00159
00160         if(!body.get("id").isEmpty()) {
00161             Optional<DialogNode> match = dnr.findById(body.get("id"));
00162             if (match.isEmpty()) {
00163                 throw new ResponseStatusException(
00164                     HttpStatus.NOT_FOUND, "couldn't find node"
00165                 );
00166             }
00167             returned.add(match.get());
00168
00169             return returned;
00170         }
00171
00172         throw new ResponseStatusException(
00173             HttpStatus.NOT_FOUND, "id cannot be empty"
00174         );
00175     }
00176
00177 }
```

17.17 thuBOT/src/main/java/com/zegline/thubot/core/controller/GUIController.java File Reference

Controller that handles the requests related to GUI display and interactions.

Classes

- class com.zegline.thubot.core.controller.GUIController

  Handles the requests related to GUI display and interactions.

Packages

- package com.zegline.thubot.core.controller

17.17.1 Detailed Description

Controller that handles the requests related to GUI display and interactions.

This controller is primarily responsible for serving the GUI pages. It fetches data from different services like the actuator information and provides it to the views.

Definition in file GUIController.java.

17.18 GUIController.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.controller;
00009
00010 import java.util.ArrayList;
00011 import java.util.List;
00012 import java.util.Map;
00013
00014 import org.springframework.beans.factory.annotation.Autowired;
00015 import org.springframework.boot.actuate.info.InfoEndpoint;
00016 import org.springframework.security.core.annotation.AuthenticationPrincipal;
00017 import org.springframework.security.core.userdetails.UserDetails;
00018 import org.springframework.security.crypto.password.PasswordEncoder;
00019 import org.springframework.stereotype.Controller;
00020 import org.springframework.ui.Model;
00021 import org.springframework.web.bind.annotation.GetMapping;
00022 import org.springframework.web.bind.annotation.PostMapping;
00023 import org.springframework.web.bind.annotation.RequestParam;
00024
00025 import com.zegline.thubot.core.model.security.User;
00026 import com.zegline.thubot.core.repository.UserRepository;
00027
00036 @Controller
00037 public class GUIController {
00038
00039     @Autowired
00040     private UserRepository ur;
00041
00042     @Autowired
00043     private PasswordEncoder passwordEncoder;
00044
00045     @Autowired
00046     private InfoEndpoint infoEndpoint;
00047
00054     @GetMapping("/")
00055     public String getIndex(Model model) {
00056         String jsonData = infoEndpoint.info().get("git").toString();
00057         model.addAttribute("jsonData", jsonData);
00058         return "index";
00059     }
00060
00066     @GetMapping("/database/display")
00067     public String getDN() {
00068         return "explore_nodes";
00069     }
00070
00076     @GetMapping("/database/display/static")
00077     public String getDN1() {
00078         return "explore_nodes_static";
00079     }
00080
00088     @GetMapping("/database/form")
00089     public String getDBEntry(Model model, @AuthenticationPrincipal UserDetails userDetails){
00090         String jsonData = infoEndpoint.info().get("git").toString();
00091         model.addAttribute("commitid", jsonData);
00092         model.addAttribute("loggedInUser", userDetails.getUsername());
00093         return "databaseForm";
00094     }
```

```
00095
00102     @GetMapping("/login")
00103     public String login(Model model) {
00104         String jsonData = infoEndpoint.info().get("git").toString();
00105         model.addAttribute("commitid", jsonData);
00106         return "login";
00107     }
00108
00115     @GetMapping("/register")
00116     public String showRegisterForm(Model model) {
00117         String jsonData = infoEndpoint.info().get("git").toString();
00118         model.addAttribute("commitid", jsonData);
00119         return "register";
00120     }
00121
00127     @GetMapping("/access-denied")
00128     public String accessDeniedPage(){
00129         return "access-denied";
00130     }
00131
00139     @PostMapping("/register")
00140     public String registerUser(Model model, @RequestParam Map<String, String> body) {
00141         List<String> errors = new ArrayList<>();
00142         String jsonData = infoEndpoint.info().get("git").toString();
00143         model.addAttribute("commitid", jsonData);
00144         String submittedUsername = body.get("username");
00145         String submittedPassword = body.get("password");
00146         String submittedPasswordConfirmed = body.get("password_confirm");
00147
00148         if (!submittedPassword.equals(submittedPasswordConfirmed)) {
00149             errors.add("Passwords do not match. Please try again.");
00150         }
00151
00152         if (ur.existsByUsername(submittedUsername)) {
00153             errors.add("User with this username already exists");
00154         }
00155
00156         if (!errors.isEmpty()) {
00157             model.addAttribute("errors", errors);
00158         } else {
00159             User u = new User();
00160             u.setUsername(submittedUsername);
00161             u.setPassword(passwordEncoder.encode(submittedPassword));
00162             ur.save(u);
00163             model.addAttribute("message", "Registration was successful. Please log in");
00164         }
00165         return "register";
00166     }
00167 }
```

17.19 thuBOT/src/main/java/com/zegline/thubot/core/controller/User↩ Controller.java File Reference

Controller for handling User-related requests.

Classes

- class com.zegline.thubot.core.controller.UserController

  Controller class for User-related actions.

Packages

- package com.zegline.thubot.core.controller

17.19.1 Detailed Description

Controller for handling User-related requests.

This controller handles the creation of User entities. However, it is meant only for testing and should not be included in a staging or production deployment.

Definition in file UserController.java.

17.20 UserController.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.controller;
00009
00010 import com.zegline.thubot.core.model.security.User;
00011 import com.zegline.thubot.core.repository.UserRepository;
00012
00013 import org.springframework.beans.factory.annotation.Autowired;
00014 import org.springframework.http.ResponseEntity;
00015 import org.springframework.web.bind.annotation.PostMapping;
00016 import org.springframework.web.bind.annotation.RequestBody;
00017 import org.springframework.web.bind.annotation.RequestMapping;
00018 import org.springframework.web.bind.annotation.RestController;
00019 import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
00020
00028 @RestController
00029 @RequestMapping("/user")
00030 public class UserController {
00031
00032     @Autowired
00033     UserRepository ur;
00034
00035     private final BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
00036
00037     // Route for creating a user
00038     // ONLY FOR TESTING
00039     // DO NOT DEPLOY IN STAGING EVER
00040     // TODO: REMOVE BEFORE DEPLOYING TO STAGING
00041     @PostMapping("/create")
00042     public ResponseEntity<User> createUser(@RequestBody User user) {
00043         String hashedPassword = passwordEncoder.encode(user.getPassword());
00044         user.setPassword(hashedPassword);
00045         User createdUser = ur.save(user);
00046         createdUser.setPassword(null);
00047         return ResponseEntity.ok(createdUser);
00048     }
00049
00050 }
```

17.21 thuBOT/src/main/java/com/zegline/thubot/core/controller/UserInputController.java File Reference

Controller for handling user input related requests.

Classes

- class com.zegline.thubot.core.controller.UserInputController

  Controller to manage user input related actions.

Packages

- package com.zegline.thubot.core.controller

17.21.1 Detailed Description

Controller for handling user input related requests.

This controller is responsible for processing user input and retrieving appropriate responses from the DialogNodeMatch service.

Definition in file UserInputController.java.

17.22 UserInputController.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.controller;
00009
00010 import org.springframework.beans.factory.annotation.Autowired;
00011 import org.springframework.beans.factory.annotation.Value;
00012 import org.springframework.web.bind.annotation.GetMapping;
00013 import org.springframework.web.bind.annotation.RequestMapping;
00014 import org.springframework.web.bind.annotation.RequestParam;
00015 import org.springframework.web.bind.annotation.RestController;
00016
00017 import com.zegline.thubot.core.model.DialogNode;
00018 import com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch;
00019
00027 @RestController
00028 @RequestMapping("/api/input")
00029 public class UserInputController {
00030
00034     @Value("${openai.api.key}")
00035     private String openaiApiKey;
00036
00040     @Autowired
00041     private DialogNodeMatch dialogNodeMatchService;
00042
00049     @GetMapping("/ask")
00050     public DialogNode inputAsk(@RequestParam String userInput) {
00051         return dialogNodeMatchService.getResponseNode(userInput);
00052     }
00053 }
```

17.23 thuBOT/src/main/java/com/zegline/thubot/core/DataLoader.java File Reference

Classes

- class com.zegline.thubot.core.DataLoader

Packages

- package com.zegline.thubot.core

17.24 DataLoader.java

Go to the documentation of this file.

```
00001 package com.zegline.thubot.core;
00002
00003 import org.springframework.boot.CommandLineRunner;
00004 import org.springframework.stereotype.Component;
00005 import org.springframework.context.annotation.Profile;
00006 import org.springframework.beans.factory.annotation.Autowired;
00007
00008 import com.zegline.thubot.core.model.DialogNode;
00009 import com.zegline.thubot.core.repository.DialogNodeRepository;
00010
00011 @Component
00012 @Profile("!test")
00013 public class DataLoader implements CommandLineRunner {
00014
00015     @Autowired
00016     private DialogNodeRepository dialogNodeRepository;
00017
00018     @Override
00019     public void run(String... args) throws Exception {
00020
00021         dialogNodeRepository.deleteAll();
```

```
00022
00023         // Root node
00024         DialogNode rootNode = new DialogNode("What can I help you with?", "Here are some options you
      might be interested in.");
00025         dialogNodeRepository.save(rootNode);
00026
00027         // Level 1 nodes
00028         DialogNode studentNode = new DialogNode("Are you a Student?", "Select your faculty or
      service.");
00029         studentNode.setParent(rootNode);
00030         studentNode = dialogNodeRepository.save(studentNode);
00031
00032         DialogNode prospectiveStudentNode = new DialogNode("Are you a Prospective Student?", "Which
      program are you interested in?");
00033         prospectiveStudentNode.setParent(rootNode);
00034         prospectiveStudentNode = dialogNodeRepository.save(prospectiveStudentNode);
00035
00036         // Level 2 nodes under 'Student'
00037         DialogNode facultyNode = new DialogNode("Which faculty?", "Here are the different
      faculties.");
00038         facultyNode.setParent(studentNode);
00039         facultyNode = dialogNodeRepository.save(facultyNode);
00040
00041         DialogNode servicesNode = new DialogNode("Looking for student services?", "Here's a list of
      services.");
00042         servicesNode.setParent(studentNode);
00043         servicesNode = dialogNodeRepository.save(servicesNode);
00044
00045         // Level 2 nodes under 'Prospective Student'
00046         DialogNode bachelorNode = new DialogNode("Interested in Bachelor's programs?", "Here are the
      Bachelor's programs we offer.");
00047         bachelorNode.setParent(prospectiveStudentNode);
00048         bachelorNode = dialogNodeRepository.save(bachelorNode);
00049
00050         DialogNode masterNode = new DialogNode("Interested in Master's programs?", "Here are the
      Master's programs we offer.");
00051         masterNode.setParent(prospectiveStudentNode);
00052         masterNode = dialogNodeRepository.save(masterNode);
00053
00054         // Level 3 nodes under 'facultyNode'
00055         DialogNode engineeringNode = new DialogNode("Engineering", "Questions related to the
      Engineering faculty.");
00056         engineeringNode.setParent(facultyNode);
00057         engineeringNode = dialogNodeRepository.save(engineeringNode);
00058
00059         DialogNode businessNode = new DialogNode("Business", "Questions related to the Business
      faculty.");
00060         businessNode.setParent(facultyNode);
00061         businessNode = dialogNodeRepository.save(businessNode);
00062
00063         // Level 3 nodes under 'servicesNode'
00064         DialogNode counselingNode = new DialogNode("Counseling Services", "Get help from Counseling
      Services.");
00065         counselingNode.setParent(servicesNode);
00066         counselingNode = dialogNodeRepository.save(counselingNode);
00067
00068         DialogNode financialAidNode = new DialogNode("Financial Aid", "Information regarding Financial
      Aid.");
00069         financialAidNode.setParent(servicesNode);
00070         financialAidNode = dialogNodeRepository.save(financialAidNode);
00071
00072         // Level 3 nodes under 'bachelorNode'
00073         DialogNode bachelorEngineeringNode = new DialogNode("Bachelor in Engineering", "Learn about
      our Engineering program.");
00074         bachelorEngineeringNode.setParent(bachelorNode);
00075         bachelorEngineeringNode = dialogNodeRepository.save(bachelorEngineeringNode);
00076
00077         DialogNode bachelorBusinessNode = new DialogNode("Bachelor in Business", "Learn about our
      Business program.");
00078         bachelorBusinessNode.setParent(bachelorNode);
00079         bachelorBusinessNode = dialogNodeRepository.save(bachelorBusinessNode);
00080
00081         // Level 3 nodes under 'masterNode'
00082         DialogNode masterEngineeringNode = new DialogNode("Master in Engineering", "Explore the
      Master's program in Engineering.");
00083         masterEngineeringNode.setParent(masterNode);
00084             masterEngineeringNode = dialogNodeRepository.save(masterEngineeringNode);
00085
00086         DialogNode masterBusinessNode = new DialogNode("Master in Business Administration", "Explore
      the MBA program.");
00087         masterBusinessNode.setParent(masterNode);
00088         masterBusinessNode = dialogNodeRepository.save(masterBusinessNode);
00089     }
00090 }
```

17.25 thuBOT/src/main/java/com/zegline/thubot/core/model/DialogNode.java File Reference

Entity representing a dialog node in a conversation flow.

Classes

- class com.zegline.thubot.core.model.DialogNode

  Represents a node within a dialog conversation.

Packages

- package com.zegline.thubot.core.model

17.25.1 Detailed Description

Entity representing a dialog node in a conversation flow.

This class is used to model a node in a conversational dialog flow, where each node represents a point in the conversation. Nodes have a hierarchical structure with parent and child relationships

Definition in file DialogNode.java.

17.26 DialogNode.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.model;
00009
00010 import java.util.HashSet;
00011 import java.util.Set;
00012
00013 import com.fasterxml.jackson.annotation.JsonIgnore;
00014
00015 import jakarta.persistence.*;
00016
00017 import lombok.AllArgsConstructor;
00018 import lombok.Builder;
00019 import lombok.Data;
00020 import lombok.NoArgsConstructor;
00021 import lombok.Getter;
00022
00023 import org.hibernate.annotations.GenericGenerator;
00024 import org.hibernate.annotations.Parameter;
00025
00033 @Data
00034 @AllArgsConstructor
00035 @NoArgsConstructor
00036 @Builder
00037 @Entity
00038 public class DialogNode {
00039
00040     @Getter
00041     @Id
00042     @GeneratedValue(generator = "questionid-generator")
00043     @GenericGenerator(name = "questionid-generator", strategy =
      "com.zegline.thubot.core.utils.generator.QuestionIdGenerator", parameters = {
00044             @Parameter(name = "prefix", value = "QN") })
00045
00046     @Column(name = "id")
00047     private String id;
00048
```

```
00049     @Column(name = "dialog_text")
00050     private String dialogText;
00051
00052     @Getter
00053     @Column(name = "msg_text")
00054     private String msgText;
00055
00056     @OneToMany(mappedBy = "dialogNode")
00057     Set<DialogNodeToResponse> questionresponse;
00058
00059     @JsonIgnore
00060     @ManyToOne
00061     @JoinColumn(name = "parent_id")
00062     private DialogNode parent;
00063
00064     @Getter
00065     @OneToMany(mappedBy = "parent", fetch = FetchType.EAGER, cascade = CascadeType.ALL)
00066     private Set<DialogNode> children = new HashSet<>();
00067
00074     public DialogNode(String dialogText, String msgText) {
00075         this.dialogText = dialogText;
00076         this.msgText = msgText;
00077     }
00078
00085     public DialogNode addChild(DialogNode c) {
00086         this.children.add(c);
00087         c.setParent(this);
00088         return this;
00089     }
00090
00097     public DialogNode addChildren(Set<DialogNode> nodes) {
00098         for (DialogNode n : nodes) {
00099             this.children.add(n);
00100             n.setParent(this);
00101         }
00102         return this;
00103     }
00104
00110     public String toString() {
00111         return "<Dialog> " + dialogText;
00112     }
00113
00118     public void setDialogText(String dialogText) {
00119         this.dialogText = dialogText;
00120     }
00121
00126     public void setMsgText(String msgText) {
00127         this.msgText = msgText;
00128     }
00129
00135     @Override
00136     public boolean equals(Object obj) {
00137         if (this == obj) return true;
00138         if (obj == null || getClass() != obj.getClass()) return false;
00139         DialogNode other = (DialogNode) obj;
00140         return id != null && id.equals(other.getId());
00141     }
00142
00147     @Override
00148     public int hashCode() {
00149         return getClass().hashCode();
00150     }
00151 }
```

17.27 thuBOT/src/main/java/com/zegline/thubot/core/model/DialogNodeToResponse.java File Reference

Entity class that represents the association between dialog nodes and responses.

Classes

- class com.zegline.thubot.core.model.DialogNodeToResponse

  Represents a link between a DialogNode and a Response entity.

Packages

- package com.zegline.thubot.core.model

17.27.1 Detailed Description

Entity class that represents the association between dialog nodes and responses.

This class is a join entity that maps the many-to-many relationship between dialog nodes and responses into two many-to-one relationships

Definition in file DialogNodeToResponse.java.

17.28 DialogNodeToResponse.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.model;
00009
00010 import jakarta.persistence.Entity;
00011 import jakarta.persistence.Id;
00012 import jakarta.persistence.JoinColumn;
00013 import jakarta.persistence.ManyToOne;
00014
00023 @Entity
00024 public class DialogNodeToResponse {
00025
00026     @Id
00027     private long id;
00028
00029     @ManyToOne
00030     @JoinColumn(name = "question_id")
00031     private DialogNode dialogNode;
00032
00033     @ManyToOne
00034     @JoinColumn(name = "response_id")
00035     private Response response;
00036
00042     public DialogNode getQuestion() {
00043         return dialogNode;
00044     }
00045
00051     public Response getResponse() {
00052         return response;
00053     }
00054 }
```

17.29 thuBOT/src/main/java/com/zegline/thubot/core/model/↩ Response.java File Reference

Entity class for storing response text associated with dialog nodes.

Classes

- class com.zegline.thubot.core.model.Response

  Entity representing a response in the system.

Packages

- package com.zegline.thubot.core.model

17.29.1 Detailed Description

Entity class for storing response text associated with dialog nodes.

This class represents a response entity in the database which stores the response text. Each response can be associated with multiple dialog nodes through a DialogNodeToResponse relationship.

Definition in file Response.java.

17.30 Response.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.model;
00009
00010 import java.util.Set;
00011
00012 import jakarta.persistence.Column;
00013 import jakarta.persistence.Entity;
00014 import jakarta.persistence.GeneratedValue;
00015 import jakarta.persistence.GenerationType;
00016 import jakarta.persistence.Id;
00017 import jakarta.persistence.OneToMany;
00018
00026 @Entity
00027 public class Response {
00028
00029     @Id
00030     @GeneratedValue(strategy=GenerationType.AUTO)
00031     private long id;
00032
00033     @Column(name = "response_text")
00034     private String response_text;
00035
00036     @OneToMany(mappedBy = "response")
00037     Set<DialogNodeToResponse> dialogNodeRespons;
00038
00042     public Response() {}
00043
00049     public Response(String r) {
00050         response_text = r;
00051     }
00052
00058     public String getResponseText() {
00059         return response_text;
00060     }
00061
00067     public void setResponseText(String rt) {
00068         response_text = rt;
00069     }
00070
00071 }
```

17.31 thuBOT/src/main/java/com/zegline/thubot/core/model/security/↩ Privilege.java File Reference

Entity representing a user privilege in user management.

Classes

- class com.zegline.thubot.core.model.security.Privilege

  Entity to represent a specific privilege.

Packages

- package com.zegline.thubot.core.model.security

17.31.1 Detailed Description

Entity representing a user privilege in user management.

This class is an entity that represents a specific privilege the user can have and is used as part of the role-based access control system.

Definition in file Privilege.java.

17.32 Privilege.java

Go to the documentation of this file.

```
00001
00007 package com.zegline.thubot.core.model.security;
00008
00009 import jakarta.persistence.*;
00010
00011 import lombok.Getter;
00012
00013 import java.util.Collection;
00014
00022 @Entity
00023 public class Privilege {
00024
00028     @Id
00029     @GeneratedValue(strategy = GenerationType.AUTO)
00030     private Long id;
00031
00035     @Getter
00036     private String name;
00037
00041     @ManyToMany(mappedBy = "privileges")
00042     private Collection<Role> roles;
00043
00049     public Privilege(String name) {
00050         this.name = name;
00051     }
00052
00056     public Privilege() {
00057     }
00058 }
```

17.33 thuBOT/src/main/java/com/zegline/thubot/core/model/security/↩ Role.java File Reference

Entity representing a user role in user management.

Classes

- class com.zegline.thubot.core.model.security.Role

  Entity to represent a specific role.

Packages

- package com.zegline.thubot.core.model.security

17.33.1 Detailed Description

Entity representing a user role in user management.

This class is an entity that represents a specific role the user can have. Each role combines multiple Privileges.

Definition in file Role.java.

17.34 Role.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.model.security;
00009
00010 import jakarta.persistence.*;
00011
00012 import lombok.Getter;
00013
00014 import java.util.Collection;
00015
00024 @Entity
00025 public class Role {
00026
00030     @Id
00031     @GeneratedValue(strategy = GenerationType.AUTO)
00032     private Long id;
00033
00037     @Getter
00038     private String name;
00039
00043     @ManyToMany(mappedBy = "roles")
00044     private Collection<User> users;
00045
00049     @Getter
00050     @ManyToMany(fetch = FetchType.EAGER)
00051     @JoinTable(
00052             name = "roles_privileges",
00053             joinColumns = @JoinColumn(
00054                     name = "role_id", referencedColumnName = "id"
00055             ),
00056             inverseJoinColumns = @JoinColumn(
00057                     name = "privilege_id", referencedColumnName = "id"
00058             )
00059     )
00060
00061     private Collection<Privilege> privileges;
00062
00068     public Role(String name) {
00069         this.name = name;
00070     }
00071
00075     public Role(){
00076     }
00077
00083     public void setPrivileges(Collection<Privilege> privileges) {
00084         this.privileges = privileges;
00085     }
00086 }
```

17.35 thuBOT/src/main/java/com/zegline/thubot/core/model/security/User.java File Reference

Entity representing a user in the user management framework.

Classes

- class com.zegline.thubot.core.model.security.User

  Entity representing a user.

Packages

- package com.zegline.thubot.core.model.security

17.35.1 Detailed Description

Entity representing a user in the user management framework.

This class is an entity that represents a user of the system. It associates each user with a username and a password. Each User can have several roles, which are used for role-based Access Control.

Definition in file User.java.

17.36 User.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.model.security;
00009
00010 import jakarta.persistence.*;
00011
00012 import lombok.Getter;
00013
00014 import java.util.Collection;
00015
00025 @Entity
00026 public class User {
00027
00031     @Id
00032     @GeneratedValue(strategy = GenerationType.AUTO)
00033     private Long id;
00034
00038     @Column(nullable = false, unique = true)
00039     @Getter
00040     private String username;
00041
00045     @Getter
00046     private String password;
00047
00051     @Getter
00052     @ManyToMany(fetch = FetchType.EAGER)
00053     @JoinTable(
00054             name = "users_roles",
00055             joinColumns = @JoinColumn(
00056                     name = "user_id", referencedColumnName = "id"
00057             ),
00058             inverseJoinColumns = @JoinColumn(
00059                     name = "role_id", referencedColumnName = "id"
00060             )
00061     )
00062
00063     private Collection<Role> roles;
```

```
00064
00070     public void setUsername(String username) {
00071         this.username = username;
00072     }
00073
00079     public void setPassword(String password) {
00080         this.password = password;
00081     }
00082
00088     public void setRoles(Collection<Role> roles) {
00089         this.roles = roles;
00090     }
00091 }
```

17.37 thuBOT/src/main/java/com/zegline/thubot/core/repository/DialogNodeRepository.java File Reference

Interface for CRUD operations on DialogNode entities.

Classes

- interface com.zegline.thubot.core.repository.DialogNodeRepository

  Interface providing CRUD operations and custom queries for DialogNode entities.

Packages

- package com.zegline.thubot.core.repository

17.37.1 Detailed Description

Interface for CRUD operations on DialogNode entities.

This interface extends the CrudRepository to provide CRUD operations for DialogNode entities. It includes additional query methods for specific search criteria related to DialogNodes, such as fetching nodes without children or finding leaf nodes within the dialog tree structure.

Definition in file DialogNodeRepository.java.

17.38 DialogNodeRepository.java

Go to the documentation of this file.

```
00001
00009 package com.zegline.thubot.core.repository;
00010
00011 import org.springframework.data.repository.CrudRepository;
00012 import org.springframework.stereotype.Repository;
00013
00014 import com.zegline.thubot.core.model.DialogNode;
00015
00016 import java.util.List;
00017
00022 @Repository
00023 public interface DialogNodeRepository extends CrudRepository<DialogNode, String> {
00024
00031     DialogNode findByChildren(DialogNode child);
00032
00038     List<DialogNode> findDialogNodesByParentIsNull();
00039 }
```

17.39 thuBOT/src/main/java/com/zegline/thubot/core/repository/DialogNodeResponseRepository.java File Reference

Interface for CRUD operations on DialogNodeToResponse entities.

Classes

- interface com.zegline.thubot.core.repository.DialogNodeResponseRepository

  Interface providing CRUD operations and custom queries for DialogNodeToResponse entities.

Packages

- package com.zegline.thubot.core.repository

17.39.1 Detailed Description

Interface for CRUD operations on DialogNodeToResponse entities.

This interface extends the CrudRepository to provide CRUD operations for DialogNodeToResponse entities. It includes an additional query method for finding associations by their DialogNode.

Definition in file DialogNodeResponseRepository.java.

17.40 DialogNodeResponseRepository.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.repository;
00009
00010 import org.springframework.data.repository.CrudRepository;
00011 import org.springframework.stereotype.Repository;
00012
00013 import com.zegline.thubot.core.model.DialogNodeToResponse;
00014 import com.zegline.thubot.core.model.DialogNode;
00015
00016 import java.util.List;
00017
00022 @Repository
00023 public interface DialogNodeResponseRepository extends CrudRepository<DialogNodeToResponse, Long> {
00024
00031     List<DialogNodeToResponse> findByDialogNode(DialogNode dialogNode);
00032 }
```

17.41 thuBOT/src/main/java/com/zegline/thubot/core/repository/PrivilegeRepository.java File Reference

Interface for CRUD operations on Privilege entities.

Classes

- interface com.zegline.thubot.core.repository.PrivilegeRepository

  Interface providing CRUD operations and custom queries for Privilege entities.

Packages

- package com.zegline.thubot.core.repository

17.41.1 Detailed Description

Interface for CRUD operations on Privilege entities.

This interface extends the JpaRepository to provide CRUD operations for Privilege entities. It includes an additional query method for finding privileges by name.

Definition in file PrivilegeRepository.java.

17.42 PrivilegeRepository.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.repository;
00009
00010 import com.zegline.thubot.core.model.security.Privilege;
00011 import org.springframework.data.jpa.repository.JpaRepository;
00012
00017 public interface PrivilegeRepository extends JpaRepository<Privilege, Long> {
00018
00025     Privilege findByName(String name);
00026 }
```

17.43 thuBOT/src/main/java/com/zegline/thubot/core/repository/↩ ResponseRepository.java File Reference

Interface for CRUD operations on Response entities.

Classes

- interface com.zegline.thubot.core.repository.ResponseRepository

  Repository interface providing CRUD operations for Response entities.

Packages

- package com.zegline.thubot.core.repository

17.43.1 Detailed Description

Interface for CRUD operations on Response entities.

This interface extends the CrudRepository to provide CRUD operations for Response entities.

Definition in file ResponseRepository.java.

17.44 ResponseRepository.java

Go to the documentation of this file.

```
00001
00007 package com.zegline.thubot.core.repository;
00008
00009 import org.springframework.data.repository.CrudRepository;
00010 import org.springframework.stereotype.Repository;
00011
00012 import com.zegline.thubot.core.model.Response;
00013
00021 @Repository
00022 public interface ResponseRepository extends CrudRepository<Response, Long> {
00023 }
```

17.45 thuBOT/src/main/java/com/zegline/thubot/core/repository/RoleRepository.java File Reference

Interface for CRUD operations on Role entities.

Classes

- interface com.zegline.thubot.core.repository.RoleRepository

  Repository interface providing CRUD operations and a method to find Role by name.

Packages

- package com.zegline.thubot.core.repository

17.45.1 Detailed Description

Interface for CRUD operations on Role entities.

This interface extends JpaRepository to provide CRUD operations for Role entities. It includes additional method to find Role by its name.

Definition in file RoleRepository.java.

17.46 RoleRepository.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.repository;
00009
00010 import com.zegline.thubot.core.model.security.Role;
00011 import org.springframework.data.jpa.repository.JpaRepository;
00012
00021 public interface RoleRepository extends JpaRepository<Role, Long> {
00022     Role findByName(String name);
00023 }
```

17.47 thuBOT/src/main/java/com/zegline/thubot/core/repository/User↩ Repository.java File Reference

Interface for CRUD operations on User entities.

Classes

- interface com.zegline.thubot.core.repository.UserRepository

  Repository interface for User entities providing CRUD operations and additional methods.

Packages

- package com.zegline.thubot.core.repository

17.47.1 Detailed Description

Interface for CRUD operations on User entities.

This interface extends JpaRepository to provide CRUD operations for User entities. It contains additional methods to find a User by its username and to check if a User with a specified username exists.

Definition in file UserRepository.java.

17.48 UserRepository.java

Go to the documentation of this file.

```
00001
00007 package com.zegline.thubot.core.repository;
00008
00009 import com.zegline.thubot.core.model.security.User;
00010 import org.springframework.data.jpa.repository.JpaRepository;
00011
00018 public interface UserRepository extends JpaRepository<User, Long> {
00019
00025     User findByUsername(String username);
00026
00032     boolean existsByUsername(String username);
00033 }
```

17.49 thuBOT/src/main/java/com/zegline/thubot/core/service/dialog↩ NodeMatch/DialogNodeMatch.java File Reference

Service Class for matching dialog nodes to user input.

Classes

- class com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatch

  Service class for matching user input to dialog nodes.

Packages

- package com.zegline.thubot.core.service.dialogNodeMatch

17.49.1 Detailed Description

Service Class for matching dialog nodes to user input.

This service class provides functionality to match user input with dialog nodes, utilizing both local repository data and external OpenAI services.

Definition in file DialogNodeMatch.java.

17.50 DialogNodeMatch.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.service.dialogNodeMatch;
00009
00010 import org.springframework.beans.factory.annotation.Autowired;
00011 import org.springframework.stereotype.Service;
00012
00013 import com.zegline.thubot.core.model.DialogNode;
00014 import com.zegline.thubot.core.repository.DialogNodeRepository;
00015 import com.zegline.thubot.core.service.openai.OpenAIService;
00016
00017 import java.util.List;
00018 import java.util.ArrayList;
00019
00028 @Service
00029 public class DialogNodeMatch {
00030
00031     @Autowired
00032     private DialogNodeRepository dialogNodeRepository;
00033
00034     @Autowired
00035     private OpenAIService openAIService;
00036
00045     public DialogNode getResponseNode(String userInput){
00046
00047         int recurseLevel = 15;
00048         DialogNode root = dialogNodeRepository.findDialogNodesByParentIsNull().get(0);
00049         DialogNode matchedNode = matchNodeToInput(userInput);
00050
00051         if (matchedNode != null) return matchedNode;
00052
00053         List<DialogNode> possibleNodes = getNodesRecursively(root, recurseLevel);
00054         List<String> possibleResponses = getAnswers(possibleNodes);
00055         List<String> responseList = openAIService.getQuestionMatch(userInput, possibleResponses);
00056
00057         if(responseList.isEmpty())
00058             return DialogNode.builder().msgText("PROMPT GOES AGAINST OUR AULA").build();
00059
00060         String unsafeNum = responseList.get(0).replace("QUESTION", "");
00061         unsafeNum = unsafeNum.replace("\"", "");
00062
00063         try {
00064             int num = Integer.parseInt(unsafeNum);
00065             return possibleNodes.get(num);
00066         } catch (Exception e) {
00067             return DialogNode.builder().msgText("PROMPT GOES AGAINST OUR AULA").build();
00068         }
00069     }
00070
00077     private List<String> getAnswers(List<DialogNode> nodes) {
00078         List<String> answers = new ArrayList<>();
00079         for (DialogNode node : nodes) {
00080             answers.add(node.getMsgText());
00081         }
00082         return answers;
00083     }
00084
00092     public static List<DialogNode> getNodesRecursively(DialogNode root, int recurseLevel) {
```

```
00093         List<DialogNode> result = new ArrayList<>();
00094         getNodesRecursivelyHelper(root, recurseLevel, result);
00095         return result;
00096     }
00097
00105     private static void getNodesRecursivelyHelper(DialogNode node, int recurseLevel, List<DialogNode>
      result) {
00106         if (node == null || recurseLevel < 0) {
00107             return;
00108         }
00109         result.add(node);
00110         node.getChildren().size();
00111
00112         if (recurseLevel > 0) {
00113             for (DialogNode child : node.getChildren()) {
00114                 getNodesRecursivelyHelper(child, recurseLevel - 1, result);
00115             }
00116         }
00117     }
00118
00125     private DialogNode matchNodeToInput(String input) {
00126         // TODO: Implement database matching logic
00127         return null;
00128     }
00129 }
```

17.51 thuBOT/src/main/java/com/zegline/thubot/core/service/openai/↩ OpenAIService.java File Reference

Service class for interactions with the OpenAI API.

Classes

- class com.zegline.thubot.core.service.openai.OpenAIService

  Service class for OpenAI API interaction.

Packages

- package com.zegline.thubot.core.service.openai

17.51.1 Detailed Description

Service class for interactions with the OpenAI API.

This class connects to the OpenAI API, sends HTTP POST requests, and processes the responses. It follows the API's specified format for requests and handles the response parsing and error handling.

Definition in file OpenAIService.java.

17.52 OpenAIService.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.service.openai;
00009
00010 import org.springframework.beans.factory.annotation.Value;
00011 import org.springframework.stereotype.Service;
00012
00013 import java.util.List;
00014 import java.net.URL;
00015 import java.net.HttpURLConnection;
00016 import java.io.DataOutputStream;
00017 import java.io.BufferedReader;
00018 import java.io.InputStreamReader;
00019 import java.util.ArrayList;
00020
00021 import com.fasterxml.jackson.databind.JsonNode;
00022 import com.fasterxml.jackson.databind.ObjectMapper;
00023
00031 @Service
00032 public class OpenAIService {
00033
00034     @Value("${openai.api.key}")
00035     private String openaiApiKey;
00036
00047      public List<String> getQuestionMatch(String input_question, List<String> list_nodes) {
00048
00049         List<String> responseList = new ArrayList<>();
00050         StringBuilder openaiInput = new StringBuilder("You will match the user input to one of the
      following questions:");
00051
00052         for (String node : list_nodes) {
00053
      openaiInput.append("QUESTION").append(list_nodes.indexOf(node)).append(":").append(node).append(";");
00054         }
00055
00056         openaiInput.append("You will ONLY respond with the Question NUMBER that you think MAKES SENSE
      to match to the input. DO NOT RESPOND WITH ANYTHING ELSE THAN QUESTIONX (where X is the number)");
00057
00058         try {
00059
00060             URL url = new URL("https://api.openai.com/v1/chat/completions");
00061             HttpURLConnection connection = (HttpURLConnection) url.openConnection();
00062             connection.setRequestMethod("POST");
00063             connection.setRequestProperty("Authorization", "Bearer " + openaiApiKey);
00064             connection.setRequestProperty("Content-Type", "application/json");
00065             connection.setDoInput(true);
00066             connection.setDoOutput(true);
00067             String requestBody = "{\"model\": \"gpt-3.5-turbo\", \"messages\": [" +
00068                     "{\"role\": \"system\", \"content\": \"" + openaiInput + "\"}," +
00069                     "{\"role\": \"user\", \"content\": \"" + input_question + "\"}" +
00070                     "], \"temperature\": 0.4, \"max_tokens\": 256, \"top_p\": 1,
      \"frequency_penalty\": 0, \"presence_penalty\": 0}";
00071
00072             try (DataOutputStream os = new DataOutputStream(connection.getOutputStream())) {
00073                 byte[] input = requestBody.getBytes("utf-8");
00074                 os.write(input, 0, input.length);
00075             }
00076
00077             int responseCode = connection.getResponseCode();
00078
00079             if (responseCode == HttpURLConnection.HTTP_OK) {
00080                 try (BufferedReader reader = new BufferedReader(new
      InputStreamReader(connection.getInputStream()))) {
00081                     StringBuilder response = new StringBuilder();
00082                     String line;
00083                     while ((line = reader.readLine()) != null) {
00084                         response.append(line);
00085                     }
00086
00087                     String jsonResponse = response.toString();
00088                     System.out.println(jsonResponse);
00089                     ObjectMapper mapper = new ObjectMapper();
00090                     JsonNode root = mapper.readTree(jsonResponse);
00091                     String finalResponse =
      root.get("choices").get(0).get("message").get("content").toString();
00092
00093                     if(finalResponse != "null") {
00094                         responseList.add(finalResponse);
00095                     }
00096                     return responseList;
00097                 }
00098             } else {
00099
```

```
00100                 System.err.println("HTTP Request failed with response code: " + responseCode);
00101                 System.out.println(requestBody);
00102             }
00103             connection.disconnect();
00104         } catch (Exception e) {
00105             e.printStackTrace();
00106         }
00107         return responseList;
00108     }
00109 }
```

17.53 thuBOT/src/main/java/com/zegline/thubot/core/service/user/ThuUserDetailService.java File Reference

Service class for handling user authentication.

Classes

- class com.zegline.thubot.core.service.user.ThuUserDetailService

  Service class for loading user-specific data needed for authentication.

Packages

- package com.zegline.thubot.core.service.user

17.53.1 Detailed Description

Service class for handling user authentication.

This class implements UserDetailsService from the Spring security framework to load user-specific data and is primarily used by the authentication manager to perform authentication.

Definition in file ThuUserDetailService.java.

17.54 ThuUserDetailService.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.service.user;
00009
00010 import com.zegline.thubot.core.model.security.User;
00011 import com.zegline.thubot.core.repository.UserRepository;
00012
00013 import org.springframework.beans.factory.annotation.Autowired;
00014 import org.springframework.security.core.userdetails.UserDetails;
00015 import org.springframework.security.core.userdetails.UserDetailsService;
00016 import org.springframework.security.core.userdetails.UsernameNotFoundException;
00017 import org.springframework.stereotype.Service;
00018
00026 @Service
00027 public class ThuUserDetailService implements UserDetailsService {
00028
00029     @Autowired
00030     private UserRepository userRepository;
00031
00039     @Override
00040     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
00041         User user = userRepository.findByUsername(username);
00042         if (user == null) {
00043             throw new UsernameNotFoundException(username);
00044         }
00045         return new ThuUserPrincipal(user);
00046     }
00047 }
```

17.55 thuBOT/src/main/java/com/zegline/thubot/core/service/user/ThuUserPrincipal.java File Reference

User principal class handling the security details of user entity.

Classes

- class com.zegline.thubot.core.service.user.ThuUserPrincipal

  Implements UserDetails interface for User authentication.

Packages

- package com.zegline.thubot.core.service.user

17.55.1 Detailed Description

User principal class handling the security details of user entity.

This class implements UserDetails interface from Spring Security and wraps around a User entity to serve as the input of authentication provider.

Definition in file ThuUserPrincipal.java.

17.56 ThuUserPrincipal.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.service.user;
00009
00010 import com.zegline.thubot.core.model.security.Privilege;
00011 import com.zegline.thubot.core.model.security.Role;
00012 import com.zegline.thubot.core.model.security.User;
00013
00014 import org.springframework.security.core.GrantedAuthority;
00015 import org.springframework.security.core.authority.SimpleGrantedAuthority;
00016 import org.springframework.security.core.userdetails.UserDetails;
00017
00018 import java.util.ArrayList;
00019 import java.util.Collection;
00020 import java.util.List;
00021
00028 public class ThuUserPrincipal implements UserDetails {
00029
00030     private User user;
00031
00037     public ThuUserPrincipal(User user) {
00038         this.user = user;
00039     }
00040
00047     @Override
00048     public Collection<? extends GrantedAuthority> getAuthorities() {
00049         return getGrantedAuthorities(getPrivileges(user.getRoles()));
00050     }
00051
00058     private List<String> getPrivileges(Collection<Role> roles) {
00059
00060         List<String> privileges = new ArrayList<>();
00061         List<Privilege> collection = new ArrayList<>();
00062         for (Role role : roles) {
00063             privileges.add(role.getName());
00064             collection.addAll(role.getPrivileges());
00065         }
```

```
00066         for (Privilege item : collection) {
00067             privileges.add(item.getName());
00068         }
00069         return privileges;
00070     }
00071
00078     private List<GrantedAuthority> getGrantedAuthorities(List<String> privileges) {
00079         List<GrantedAuthority> authorities = new ArrayList<>();
00080         for (String privilege : privileges) {
00081             authorities.add(new SimpleGrantedAuthority(privilege));
00082         }
00083         return authorities;
00084     }
00085
00091     @Override
00092     public String getPassword() {
00093         return user.getPassword();
00094     }
00095
00101     @Override
00102     public String getUsername() {
00103         return user.getUsername();
00104     }
00105
00111     @Override
00112     public boolean isAccountNonExpired() {
00113         return true;
00114     }
00115
00121     @Override
00122     public boolean isAccountNonLocked() {
00123         return true;
00124     }
00125
00131     @Override
00132     public boolean isCredentialsNonExpired() {
00133         return true;
00134     }
00135
00141     @Override
00142     public boolean isEnabled() {
00143         return true;
00144     }
00145 }
```

17.57 thuBOT/src/main/java/com/zegline/thubot/core/utils/generator/↩ QuestionIdGenerator.java File Reference

Custom identifier generator for Hibernate entities.

Classes

- class com.zegline.thubot.core.utils.generator.QuestionIdGenerator

  Custom ID generator for Hibernate entities.

Packages

- package com.zegline.thubot.core.utils.generator

17.57.1 Detailed Description

Custom identifier generator for Hibernate entities.

This class implements a custom ID generator for entities in Hibernate. It generates unique identifiers by combining a specified prefix with a random number. The uniqueness of the ID is ensured by checking against existing entities in the database

Definition in file QuestionIdGenerator.java.

17.58 QuestionIdGenerator.java

Go to the documentation of this file.

```
00001
00009 package com.zegline.thubot.core.utils.generator;
00010
00011 import java.io.Serializable;
00012
00013 import java.util.Properties;
00014 import java.util.Random;
00015
00016 import org.hibernate.MappingException;
00017 import org.hibernate.engine.spi.SharedSessionContractImplementor;
00018 import org.hibernate.id.IdentifierGenerator;
00019 import org.hibernate.service.ServiceRegistry;
00020 import org.hibernate.type.Type;
00021
00029 public class QuestionIdGenerator implements IdentifierGenerator {
00030
00031     private String prefix;
00032
00043     @Override
00044     public Serializable generate(SharedSessionContractImplementor session, Object object) {
00045         String generatedId;
00046
00047         do {
00048             int randomNum = new Random().nextInt(9000) + 1000;
00049             generatedId = prefix + randomNum;
00050         } while (isIdExists(session, generatedId));
00051         return generatedId;
00052     }
00053
00063     private boolean isIdExists(SharedSessionContractImplementor session, String generatedId) {
00064         // Check if an entity with the given ID already exists in the database
00065         return session.createQuery("select count(*) from DialogNode where id = :id", Long.class)
00066                 .setParameter("id", generatedId)
00067                 .uniqueResult() > 0;
00068     }
00069
00080     @Override
00081     public void configure(Type type, Properties params, ServiceRegistry serviceRegistry) throws
      MappingException {
00082         setPrefix(params.getProperty("prefix"));
00083     }
00084
00090     public void setPrefix(String p) {
00091         prefix = p;
00092     }
00093 }
00094
```

17.59 thuBOT/src/main/java/com/zegline/thubot/ThuBotApplication.java File Reference

Entry point for the ThuBot chatbot application.

Classes

- class com.zegline.thubot.ThuBotApplication

  Main application class for the ThuBot chatbot.

Packages

- package com.zegline.thubot

17.59.1 Detailed Description

Entry point for the ThuBot chatbot application.

This class serves as the main class. It initializes and configures the Spring Boot application framework, which facilitates dependency injection, embedded server configuration

Definition in file ThuBotApplication.java.

17.60 ThuBotApplication.java

Go to the documentation of this file.

```
00001
00009 package com.zegline.thubot;
00010
00011 import org.springframework.boot.SpringApplication;
00012 import org.springframework.boot.autoconfigure.SpringBootApplication;
00013
00021 @SpringBootApplication
00022 public class ThuBotApplication {
00023
00031     public static void main(String[] args) {
00032         SpringApplication.run(ThuBotApplication.class, args);
00033     }
00034 }
```

17.61 thuBOT/src/test/java/com/zegline/thubot/core/apiApplication↩ Tests.java File Reference

Test class for the main application context.

Classes

- class com.zegline.thubot.core.ApiApplicationTests

  Test class for the main application context.

Packages

- package com.zegline.thubot.core

17.61.1 Detailed Description

Test class for the main application context.

This class contains unit tests for the main application context of the ThuBot application. It tests the load and initialization of the application context.

Definition in file apiApplicationTests.java.

17.62 apiApplicationTests.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core;
00009
00010 import com.zegline.thubot.ThuBotApplication;
00011
00012 import org.junit.jupiter.api.Test;
00013
00014 import org.springframework.boot.test.context.SpringBootTest;
00015
00022 @SpringBootTest(classes = ThuBotApplication.class)
00023 class ApiApplicationTests {
00024
00030     @Test
00031     void contextLoads() {
00032     }
00033 }
```

17.63 thuBOT/src/test/java/com/zegline/thubot/core/model/DialogNode↩ Tests.java File Reference

Test class for the DialogNode entity.

Classes

- class com.zegline.thubot.core.model.DialogNodeTests

  Test class for the DialogNode.

Packages

- package com.zegline.thubot.core.model

17.63.1 Detailed Description

Test class for the DialogNode entity.

This class contains unit tests for the DialogNode entity. It tests the functionality of creating a DialogNode object and adding child nodes to a parent node.

Definition in file DialogNodeTests.java.

17.64 DialogNodeTests.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.model;
00009
00010 import org.junit.jupiter.api.BeforeEach;
00011 import org.junit.jupiter.api.Test;
00012
00013
00014 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
00015 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
00016 import org.springframework.test.context.ActiveProfiles;
00017
00018 import java.util.Set;
00019
00020 import static org.junit.Assert.assertEquals;
00021 import static org.junit.Assert.assertTrue;
00022
00030 @DataJpaTest()
00031 @AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
00032 @ActiveProfiles("test")
00033 public class DialogNodeTests {
00034
00035     private DialogNode dialognode;
00036
00042     @BeforeEach
00043     public void setup() {
00044         dialognode = new DialogNode("Question", "Response");
00045     }
00046
00052     @Test
00053     public void testDialogNodeCreation() {
00054         assertEquals("Question", dialognode.getDialogText());
00055         assertEquals("Response", dialognode.getMsgText());
00056     }
00057
00063     @Test
00064     public void testAddChild() {
00065         DialogNode childNode = new DialogNode("ChildQuestion", "ChildResponse");
00066         dialognode.addChild(childNode);
00067
00068         assertTrue(dialognode.getChildren().contains(childNode));
00069         assertEquals(dialognode, childNode.getParent());
00070     }
00071
00077     @Test
00078     public void testAddChildren() {
00079         DialogNode child1 = new DialogNode("Child1Question", "Child1Response");
00080         DialogNode child2 = new DialogNode("Child2Question", "Child2Response");
00081
00082         dialognode.addChildren(Set.of(child1, child2));
00083
00084         assertTrue(dialognode.getChildren().contains(child1));
00085         assertTrue(dialognode.getChildren().contains(child2));
00086         assertEquals(dialognode, child1.getParent());
00087         assertEquals(dialognode, child2.getParent());
00088     }
00089 }
```

17.65 thuBOT/src/test/java/com/zegline/thubot/core/repository/DialogNodeRepositoryTests.java File Reference

Test class for the DialogNodeRepository.

Classes

- class com.zegline.thubot.core.repository.DialogNodeRepositoryTests

  Test class for the DialogNodeRepository.

Packages

- package com.zegline.thubot.core.repository

17.65.1 Detailed Description

Test class for the DialogNodeRepository.

This class contains unit tests for methods in the DialogNodeRepository. It tests the ability to create and retrieve DialogNode instances and ensure the correct management of relationships between nodes.

Definition in file DialogNodeRepositoryTests.java.

17.66 DialogNodeRepositoryTests.java

Go to the documentation of this file.

```
00001
00009 package com.zegline.thubot.core.repository;
00010
00011 import com.zegline.thubot.core.model.DialogNode;
00012
00013 import org.junit.jupiter.api.Assertions;
00014 import org.junit.jupiter.api.Test;
00015
00016 import org.springframework.beans.factory.annotation.Autowired;
00017 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
00018 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
00019 import org.springframework.test.context.ActiveProfiles;
00020
00021 import java.util.HashSet;
00022
00030 @DataJpaTest()
00031 @AutoConfigureTestDatabase(replace = AutoConfigureTestDatabase.Replace.NONE)
00032 @ActiveProfiles("test")
00033 public class DialogNodeRepositoryTests {
00034
00035     @Autowired
00036     DialogNodeRepository dnr;
00037
00044     @Test
00045     public void DialogNode_Find_ReturnFoundDialogNode(){
00046
00047         DialogNode testNode = DialogNode.builder()
00048                 .id("QR1923")
00049                 .dialogText("This is a test")
00050                 .msgText("This is test message").build();
00051
00052         DialogNode savedDialogNode = dnr.save(testNode);
00053         Assertions.assertNotNull(savedDialogNode);
00054         Assertions.assertEquals(savedDialogNode, testNode);
00055     }
00056
00063     @Test
00064     public void DialogNodeRelationship_Find_ReturnFoundRelationship(){
00065
00066         DialogNode leafNode = DialogNode.builder()
00067                 .id("QR1000")
00068                 .dialogText("This is a leaf node")
00069                 .msgText("Leaf")
00070                 .children(new HashSet<>(1))
00071                 .build();
00072
00073         DialogNode rootNode = DialogNode.builder()
00074                 .id("QR0000")
00075                 .dialogText("This is the root node")
00076                 .msgText("Root")
00077                 .children(new HashSet<>(1))
00078                 .build();
00079         rootNode.addChild(leafNode);
00080
00081         DialogNode savedRoot = dnr.save(rootNode);
00082         DialogNode savedLeaf = dnr.save(leafNode);
00083         Assertions.assertNotNull(savedRoot.getChildren());
00084     }
00085 }
```

17.67 thuBOT/src/test/java/com/zegline/thubot/core/service/dialogNode↩ Match/DialogNodeMatchTests.java File Reference

Test class for the DialogNodeMatch service.

Classes

- class com.zegline.thubot.core.service.dialogNodeMatch.DialogNodeMatchTests

Packages

- package com.zegline.thubot.core.service.dialogNodeMatch

17.67.1 Detailed Description

Test class for the DialogNodeMatch service.

This class contains unit tests for the DialogNodeMatch service. It tests the functionality of getting the correct response node based on the input and the interaction with the OpenAI service.

Definition in file DialogNodeMatchTests.java.

17.68 DialogNodeMatchTests.java

Go to the documentation of this file.

```
00001
00008 package com.zegline.thubot.core.service.dialogNodeMatch;
00009
00010 import com.zegline.thubot.core.model.DialogNode;
00011 import com.zegline.thubot.core.repository.DialogNodeRepository;
00012 import com.zegline.thubot.core.repository.DialogNodeResponseRepository;
00013 import com.zegline.thubot.core.service.openai.OpenAIService;
00014
00015 import org.junit.jupiter.api.BeforeEach;
00016 import org.junit.jupiter.api.Test;
00017 import org.junit.jupiter.api.extension.ExtendWith;
00018 import org.mockito.InjectMocks;
00019 import org.mockito.Mock;
00020 import org.mockito.junit.jupiter.MockitoExtension;
00021
00022 import static org.mockito.ArgumentMatchers.anyList;
00023 import static org.mockito.ArgumentMatchers.anyString;
00024 import static org.mockito.Mockito.*;
00025 import static org.junit.jupiter.api.Assertions.*;
00026
00027 import java.util.*;
00028
00029 @ExtendWith(MockitoExtension.class)
00030 public class DialogNodeMatchTests {
00031
00032     @Mock
00033     private DialogNodeRepository dialogNodeRepository;
00034
00035     @Mock
00036     private DialogNodeResponseRepository dialogNodeResponseRepository;
00037
00038     @Mock
00039     private OpenAIService openAIService;
00040
00041     @InjectMocks
00042     private DialogNodeMatch dialogNodeMatch;
00043
00044     private DialogNode rootNode;
```

```
00045
00051     @BeforeEach
00052     public void setup() {
00053         rootNode = new DialogNode("RootNode", "RootResponse");
00054
      when(dialogNodeRepository.findDialogNodesByParentIsNull()).thenReturn(Collections.singletonList(rootNode));

00055     }
00056
00063     @Test
00064     public void testGetResponseNodeWithNoDirectMatchAndOpenAIProvidesValidIndex() {
00065
00066         List<String> openAIResponses = Collections.singletonList("0");
00067         when(openAIService.getQuestionMatch(anyString(), anyList())).thenReturn(openAIResponses);
00068         DialogNode resultNode = dialogNodeMatch.getResponseNode("SomeInput");
00069         assertEquals(rootNode, resultNode, "The returned node should match the root node.");
00070     }
00071
00078     @Test
00079     public void testGetResponseNodeWithOpenAIFailure() {
00080
00081         when(openAIService.getQuestionMatch(anyString(),
      anyList())).thenReturn(Collections.emptyList());
00082         DialogNode resultNode = dialogNodeMatch.getResponseNode("SomeInputWithOpenAIFailure");
00083         assertNotNull(resultNode, "A DialogNode instance should be returned even if OpenAI provides no
      match.");
00084         assertEquals("PROMPT GOES AGAINST OUR AULA", resultNode.getMsgText(), "The returned DialogNode
      should have the fallback msgText.");
00085     }
00086 }
```