Dictionary English - Tamil

Project Developed by **P. SARAVANAN**

Content:

- Requirement Documentation.
- Software Design Documentation.
- Technical Documentation (Backend coding).
- User Guide Documentation.

Project Overview:

Main Language: Java

Web Framework: Spring Boot

Tools: SpringToolSuite4

Project Purpose:

English to Tamil dictionary web application is to provide users with one or multiple meanings for a single English word, enhancing language comprehension and aiding communication between English and Tamil speakers.

Requirement Documentation

Requirements:

- Firstly, we require an English to Tamil dictionary file in text format. This web application aims to provide meanings for 56,862 words.

- The user will input an English word.

Software Design Documentation

This project comprises totally 3 web pages.

HTML, CSS,Java Script,JSP are used in this project.

Homepage and Dictionary English-Tamil(Page 1):

- **HTML Functionalities:**
- Created a bold title "Dictionary" for this page
- Created a subtitle `<h1>` in bold for "English to Tamil".
- Inserted a placeholder "Enter English Word" to guide and ensure the entry of English words only.
- Included a button labeled "Find Meanings" to search for Tamil meanings.

CSS Functionalities:

- Ensured proper centering of the page content.
- Styled the title "Dictionary" with a big bold font, black color, and added some shadows behind to present 3D letters.
- Created a corner oval box at the center of the page and designed a slightly oval input field.
- Added a hint of grey color to the background.
- Added a green color button that changes to dark green when the user hovers over it with the cursor.
- At the bottom, added a "Disclaimer" message with brown color fonts

Meanings Found page(Page 2):

JSP ,HTML,JS and CSS Functionalities:

- Created a bold title “English to Tamil Dictionary” with black color.

- Created two fields: “English word” with light blue font color for the user searched English word, and “Tamil Meanings” with green font color for the Tamil meanings of the user's input. These are dynamic fields.
- Added a “Back” button with a blue color that changes to dark blue when the user hovers over it with the cursor. This button is useful for returning from the current page. It is created using JavaScript.
- Finally, added a “Disclaimer” message to inform users about this application with brown color font.

Meanings not Found page(Page 3):

JSP ,HTML,JS and CSS Functionalities:

- Created a bold title “English to Tamil Dictionary” with black color.

- Created two fields: “English word” with light blue font color for the user searched English word. This Field are dynamic field.

- Added a "not found" message for the user. In Tamil: ">> மன்னிக்கவும், உங்கள் தேடலுக்கு பொருந்தக்கூடிய உள்ளடக்கம் எதுவும் கிடைக்கவில்லை. <<" and in English: ">> Sorry, no content found matching your search. <<" with bold red font color.

- Added a “Back” button with a blue color that changes to dark blue when the user hovers over it with the cursor. This button is useful for returning from the current page. It is created using JavaScript.
- Finally, added a “Disclaimer” message to inform users about this application with brown color font.

Technical Documentation

Created Totally 5 classes for code to "Dictionary English-Tamil".

1. An "Input_collect" class has been created to store user inputs.

2. The "Eng_Tamil_Dictionary" class finds the Tamil meanings of user-entered English words by utilizing File IO to access a text format file containing the Tamil Dictionary.

3. The "UserInput" interface is utilized to save English words inputted by users to the Database.

4. Spring Boot automatically generated the "Dictionary_English2Tamil" class.

5. All classes are linked together within the "Dictionary_Controller" (Controller Class).

Maven Dependencies:

- **Apache Tomcat Jasper:** Provides JSP (JavaServer Pages) and JSTL (JavaServer Pages Standard Tag Library) support.
- **Spring Boot Starter Data JPA**: Starter for utilizing Spring Data JPA, simplifying data access with JPA (Java Persistence API).
- **Spring Boot Starter Web:** Starter for developing web applications using Spring MVC (Model-View-Controller).
- **H2 Database:** Embedded relational database often used for development and testing purposes.
- **Spring Boot Starter Test:** Starter for testing Spring Boot applications, supporting unit and integration testing.

Here clear details about all the class code,

Package Name is "dictionary".

- This package contains classes related to the English to Tamil dictionary application.

Class Input_collect:

Description:
This class serves as a Plain Old Java Object (POJO) representing the input collected from the user(Single English Word)

Annotations

- `@Entity`: This annotation is from the Jakarta Persistence API (JPA) and is used to specify that this class is an entity mapped to a database table.
- `@Component`: This annotation is from the Spring Framework and marks this class as a Spring-managed component.

Field:

`engWord`: A private field of type `String` representing the English word input by the user. This field is annotated with `@Id`, indicating that it is the primary key of the entity.

Methods:

Getter, Setter Methods:

- `getEngWord()`: A getter method for retrieving the value of the `engWord` field.
- `setEngWord(String engWord)`: A setter method for setting the value of the `engWord` field.

Conclusion:

- The `Input_collect` class represents user input within the English to Tamil dictionary application, facilitating storage and retrieval of English words. It integrates with the Spring Framework for dependency injection and with JPA for database mapping.

Class `Eng_Tamil_Dictionary`:

Description: This class is the core component of the English to Tamil dictionary application, responsible for finding the meanings of English words in a Tamil dictionary text file.

Annotations:

- `@Component`: This annotation is from the Spring Framework and marks this class as a Spring-managed component.

Methods:

- `tamil_dictionary(Input_collect userInput)`: This method takes an `Input_collect` object representing the user input and returns a `ModelAndView` object. It reads a Tamil dictionary file, searches for the meaning of the input English word, and prepares the appropriate response based on whether the meaning is found or not.

Variables

- `notfound`: A string containing the message to display when the word is not found in the dictionary.
- `input`: A `File` object representing the Tamil dictionary file.
- `result`: A string to store the found meaning(s) of the English word.
- `word`: A string representing the English word input by the user.
- `regex`: A string containing a regular expression pattern to match Tamil words.

Flow:

1. Read the English word from the user input and convert it to lowercase.
2. Open the Tamil dictionary file for reading.
3. Iterate through each line of the dictionary file.
4. Match each line against the regular expression pattern to identify Tamil words.

5. If a match is found and the first characters match, check if the English word exists in the line followed by a tab character.
6. If found, append the matching line to the result string.
7. Close the BufferedReader after reading all lines from the dictionary file.
8. If a matching result is found, add the result and user input to the ModelAndView object and set the view name to display the found result.
9. If no meaning is found, add the appropriate message and user input to the ModelAndView object and set the view name to display the not found message.
10. Return the ModelAndView object.

Conclusion:

The `Eng_Tamil_Dictionary` class provides the core functionality of the English to Tamil dictionary application. It efficiently searches for the meanings of English words in a Tamil dictionary and prepares the appropriate response for display to the user.

`Interface UserInput :`

Description:
This interface serves as the repository for storing user input, specifically English words typed by the user.

Annotations

`@Component`: This annotation is from the Spring Framework and marks this interface as a Spring-managed component.

Extends

- `CrudRepository<Input_collect, String>`: This interface extends the `CrudRepository` interface provided by Spring Data JPA. It defines CRUD (Create, Read, Update, Delete) operations for the `Input_collect` entity, using `String` as the type of the primary key.

Methods

No additional methods: Since `UserInput` extends `CrudRepository`, it inherits all the methods for CRUD operations from the `CrudRepository` interface. These methods include `save()`, `findById()`, `findAll()`, `deleteById()`, and others. Therefore, no additional methods need to be declared in this interface.

Conclusion

The `UserInput` interface, annotated with `@Component`, extends `CrudRepository<Input_collect, String>`, enabling basic CRUD operations for storing and managing user input, specifically English words typed by the user.

Class `DictionaryEnglish2TamilApplication` :

Description:

Annotations

- `@SpringBootApplication`: This annotation is from Spring Boot and indicates that this class is the primary configuration class for the Spring Boot application. It enables auto-configuration, component scanning, and other features provided by Spring Boot.

Method

- `main(String[] args)`: This method is the starting point of the application. It uses the `SpringApplication.run()` method to launch the Spring Boot application, passing the class `DictionaryEnglish2TamilApplication` and any command-line arguments provided.

Conclusion:

The `DictionaryEnglish2TamilApplication` class, annotated with `@SpringBootApplication`, serves as the entry point for the English to Tamil dictionary application. It initiates the Spring Boot application and triggers the auto-configuration process, enabling the application to start and run smoothly.

Class `Dictionary_Controller`

Description:
This class serves as the controller for handling HTTP requests and managing the flow of data within the English to Tamil dictionary application

Dependencies

- `Eng_Tamil_Dictionary Etd`: Autowired instance of `Eng_Tamil_Dictionary` for performing dictionary operations.
- `UserInput usi`: Autowired instance of `UserInput` for data persistence.

Annotations

- `@Controller`: This annotation is from Spring MVC and marks this class as a controller component, allowing it to handle HTTP requests.

Methods

- `dictionary_Homepage()`: This method handles GET requests for the homepage of the dictionary. It returns the name of the HTML template for the dictionary homepage.

- `Meanings(Input_collect results)`: This method handles GET requests for finding meanings of input words. It calls the `tamil_dictionary()` method of the `Eng_Tamil_Dictionary` instance to find meanings and returns a `ModelAndView` object containing the result. Additionally, it saves the input data using the `save()` method of the `UserInput` instance for further analysis or reference.

Conclusion:

The `Dictionary_Controller` class, annotated with `@Controller`, serves as the core controller component for the English to Tamil dictionary application. It handles HTTP requests, delegates dictionary operations to the `Eng_Tamil_Dictionary` component, and manages data persistence using the `UserInput` component. This class plays a pivotal role in orchestrating the application's functionality and ensuring seamless user interaction.

User Documentation

How To Use This Web Application:

- First, run the “Dictionary_English2Tamil” application.
- Then navigate to “localhost:8080/tamildictionary” in your browser; you will be directed to the Home Page of the English-Tamil Dictionary web application.
- Enter the English word for which you want to find the Tamil meaning.
- You will receive the meanings of the English word present in this web application, or it will show a “not found” message to you.
- If you are on the "Meanings found" or "not found" page, there is a "Back" button available at the bottom to get back from the current page.

Note:

This web application provides meanings for 56,862 words only.