# AutoGeneralAPI application Summary

AutoGeneralAPI application is a spring boot application which exposes certain REST APIs. This application provides services like

1. Validating(each open bracket should have a closing) brackets in a string
2. Creating a new TODOItem
3. Editing a TODOItem
4. Viewing the TODOItem
5. IntegrationTesting the TODOItem endpoints

* Since spring boot provides Rapid Application development, it’s easily deployable without any requirement of web application servlet containers, being a standalone java application.
* OAuth2 is implemented for achieving security here. The Authorization Server provides an access token which can be set in the header while calling a restricted resource which is hosted in Resource Server.
* Spring Data JPA is used to write the Data access layer.
* This project utilizes embedded database (HSQL) to connect and query data. There is a customized create-table.sql written which when executed will create the table structure in the embedded database.

Thread safety for this application is achieved using below points.

1. In the controller class , ‘class’ level variables has been reduced as much as possible, since controllers can be shared by multiple threads and hence multiple threads can access this variable and change, causing stale value.
2. POJO in this application do not use instance variables to store state, to avoid inconsistent situations where concurrent threads can access a single instance of the web service class.
3. The controller class is annotated with @RequestScope, so the controller class will be accessible from single thread which is allocated to handle the request.
4. Not using static members in the objects created, since, they will be shared across threads.
5. The transaction management has been implemented for all the API resources in the application with @Transactional along with Propagation Properties.

**Endpoints**

Below are the endpoints defined in the project.

/tasks

/todo

/integrationTest

# API: Validate Brackets

*This api provides service to validate brackets in the passed input*

Endpoint: /tasks/validateBrackets

Method: GET

Data Params : { "inputString": [String]

}

Response Codes: Success (200 OK), Bad Request (400), Unauthorized (401)

## API : Create New ToDoItem

*This api provides service to create a new ToDoItem*

Endpoint: /todo

Method: POST

Data Params : { " {

"id": 1,

"text": "Uulwi ifis halahs gag erh'ongg w'ssho",

"createdAt": "2017-10-13T01:50:58.735Z",

"completed": true

}

Response Codes: Success (200 OK), Bad Request (400), Unauthorized (401)

# API : Edit a ToDoItem

*This api provides service to edit an existing ToDOItem*

Endpoint: /todo/[?id=1](http://localhost:8080/autogeneral/todo/?id=1)

Method: PATCH

Request Params : { “id” : [Long]

}

Response Codes: Success (200 OK), Bad Request (400), Unauthorized (401)

# API : View a ToDoItem

*This api provides service to view an existing ToDoItem*

Endpoint: /todo/?id=1

Method: GET

Data Params : { “id” : [Long]

}

Response Codes: Success (200 OK), Bad Request (400), Unauthorized (401)

# API : Integration Test

*This api provides service to test all the apis together***.**

Endpoint: /integrationTest/

Method: GET

URL Params : Required: apiUrl=[String]

Response Codes: Success (200 OK), Bad Request (400), Unauthorized (401)

# Application Flow

AutoGeneralAPI utilizes the spring mvc as the framework which consists of the below 4 layers. However, there is no View part associated with this application since it concentrates only on the backend service endpoints.

* Controller
* Service/ServiceImpl
* Business Object/Business Object Impl
* Repository

The controller class is annotated with @RestController, which exposes the endpoints and acts as an entry point which receive request from clients. Controller class defines the 3 endpoints namely

*/tasks – is to validate brackets in a string*

*/todo – is to create/edit and view ToDoItem*

*/integrationTest – is to test all the api*

* A browser/client request gets mapped to the controller through respective endpoint. The controller identifies the request and forwards the same to the corresponding Service layer. The code now needs to execute some business logics for which the Business classes are written and at last to talk to the database it gets connected with the Repository layer. The model/VO stores the value objects which can then be used in various layers mentioned above

The JPATransactionManager have been used to support and manage our transactions. The required tables are created in the ‘In memory’ database (HSQL), We can see and execute the queries in tables.

For security, the OAuth2 tokenization mechanism has been implemented. Here the application defines its own Authorization server using annotation @EnableAuthorizationServer. The case is same with resource server making use of @EnableResourceServer. Each of the endpoints is defined with restricted use and user requires to be authenticated to access them.

The list of users for application authentication purpose has been maintained as in memory

**Testing the Endpoints**

The APIs are tested using Postman Client.