Cognizant Academy

truYum

FSE Spring RESTful Specification Document

Version 1.0

	Prepared By / Last Updated By	Reviewed By	Approved By
Name	Chandrasekaran Janardhanan	Vimalathithan Krishnan	Ramadevanahalli Lingachar, Shashidhara Murthy
Role	Learning Solution Designer	Learning Solution Architect	Learning Solution Lead
Signature			
Date			

Table of Contents

1.0 Introduction

1.1 Purpose of this document

The purpose of this document is to define the user interface specification for truYum project.

1.2 Definitions & Acronyms

Definition / Acronym	Description
REST	Representational State Transfer

1.3 Project Overview

Refer truyum-user-stores.xlsx for understanding the functionality and features.

1.4 Scope

Develop RESTful Web Service that will be consumed by Angular

Application for truYum application.

1.5 Intended Audience

- Product Owner
- Scrum Master
- Application Architect
- Project Manager
- Test Manager
- Development Team
- Testing Team

1.6 Hardware and Software Requirement

1. Hardware Requirement:
    a. Developer Desktop PC with 8GB RAM
2. Software Requirement

a. Eclipse
b. Postman
c. JMeter
d. Visual Studio Code
e. Angular

2.0 Project Setup in Eclipse

There is no need for forking or cloning, the development for RESTful Web Service module can be done in the existing truyum-v2 project itself.

In the existing truyum-v2 project, the Eclipse project for RESTful Web Service has to be created in the ‘monolithic-service’ folder. Follow steps below to incorporate the same.

1. Open Eclipse
2. File > New > Project > Maven > Maven Project > Next
3. Check 'Create a simple project'
4. Uncheck 'Use default Workspace location'
5. Click browse and select the truyum-v2\monolithic-services folder and click Next
6. Enter Group Id as 'com.cognizant'
7. Enter Artifact Id as 'truyum'
8. Click Finish
9. If error messages are displayed click OK
10. Copy pom.xml from the spring-learn project and overwrite the pom.xml of the newly created project
11. Modify the following properties in pom.xml, so that it reflects truYum project details:

    ```
    <artifactId>truyum</artifactId>
    <name>truyum</name>
    <description>truyum web services</description>
    ```

12. Update the maven configuration by right clicking on truyum project > Maven > Update Project > OK
13. In the root folder of the newly created project include .gitignore file with the following content, which will ensure that unwanted files are not pushed to git:

    ```
        /.settings
    ```

```
/build
.classpath
.project
/bin/
/target/
```

14. Create following source files (refer code from spring-learn):

    a. com.cognizant.truyum.TruyumApplication (Spring Boot application class, refer SpringLearnApplication.java in spring-learn application, include annotations and main method from this source file)

    b. com.cognizant.truyum.WebConfig

    c. com.cognizant.truyum.exception.GlobalExceptionHandler

    d. src/main/resources/application.properties – including configuration for logs and server port.

15. Change the logger pattern as specified below, which provides improved readability of logs. Change this configuration in application.properties:

```
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-
25.25logger{25} %25M %4L %m%n
```

3.0 Environment configuration in Angular

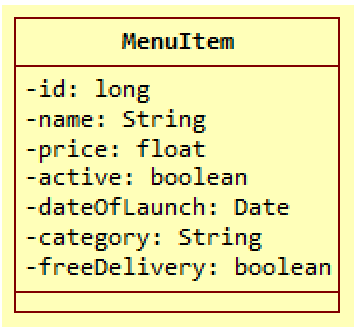
When doing hands on we were hard coding the base url of the Rest API. It is better to assign it as environment variable and refer it in service. Refer the link below and incorporate the same in the truyum angular application.

https://medium.com/@balramchavan/separating-production-and-development-http-urls-using-environment-ts-file-in-angular-4c2dd0c5a8b0

4.0 TYUS003 – View Menu Items (anonymous user)

4.1 Rest API

com.cognizant.truyum.model.MenuItem - Create class com.cognizant.truyum.model.MenuItem



src/main/resources/truyum.xml - Include bean definition to load the sample data with list of menu items. Refer implemented screen for sample data

com.cognizant.truyum.dao.MenuItemDao – Copy this interface from Java module

com.cognizant.truyum.dao.MenuItemDaoCollectionImpl

- Copy this class from Java module
- Instead of hard coding the Menu Item list, load it from the spring xml configuration file truyum.xml

com.cognizant.truyum.service.MenuItemService

- Autowire MenuItemDao with MenuItemDaoCollectionImpl
- Include method getMenuItemListCustomer() which invokes getMenuItemListCustomer() from dao.

com.cognizant.truyum.controller.MenuItemController

- Include @RequestMapping("/menu-items") at class level
- Autowire MenuItemService
- Implement method getAllMenuItems() that invokes getMenuItemListCustomer() and returns the menu item list

com.cognizant.truyum.security.JwtAuthorizationFilter

- Reuse existing code from the spring-learn example.

com.cognizant.truyum.SecurityConfig

- configure(AuthenticationManagerBuilder auth)
    - Include in memory users:
        - Username: user; Password: pwd; Role: USER
        - Username: admin; Password: pwd; Role: ADMIN
- configure(HttpSecurity httpSecurity)
    - Include CORS configuration in http security
    - Allow anonymous access to /menu-items. Refer code below:

```
httpSecurity.cors();
httpSecurity.csrf().disable().httpBasic().and()
    .authorizeRequests()
    .antMatchers("/menu-items").permitAll()
    .anyRequest().authenticated();
    .and()
    .addFilter(new JwtAuthorizationFilter(authenticationManager()));
```

com.cognizant.truyum.WebConfig - Include CORS configuration to support all method types and all resources

com.cognizant.springlearn.security.JwtAuthorizationFilter

Copy code from spring learn application for this class as it is.

4.2 Angular

Menu Item Service

- Create service in services folder
- Include method getAllMenuItems() to invoke the rest api '/menu-items' and return the menu items.

Menu Item List Component

- ngOnInit()
    - Inject MenuItemService and invoke getAllMenuItems()
    - Set the response in the item list property of the component which will reflect the data retrieved from rest api in the browser

5.0 TYUS004 – Search Food Item

Since the search food item is implemented as part of angular component along, it is sufficient to check if search works correctly.

6.0 TYUS005 – Add Item to Cart (Anonymous User)

6.1 Angular

Menu Item Info Component

- onSubmit()
    - Check if user is logged in using UserAuthService
    - If user not logged in
        - Store the selected menu item id in UserAuthService (create necessary property, getter and setter methods in UserAuthService)
        - Route to login component

7.0 TYUS006 – Login (Customer & Admin)

7.1 Rest API

com.cognizant.truyum.controller.AuthenticationController

Copy and paste AuthenticationController from angular-learn application

com.cognizant.truyum.SecurityConfig

Include authorization for '/authentication' to be allowed for both the roles USER and ADMIN

7.2 Angular

Authentication Service

- Get the base URL from environment
- Create this service based on the existing code available in angular-learn application

Login Component

- Refer login component of angular-learn application and implement similar flow and logic here.
- After successful login display menu item list component
- ngOnInit()
  - If the menu item id is present in UserAuthService then display message that they have to login first before adding item to cart.

Navigation Component

- constructor
  - Inject UserAuthService
- ngOnInit()
  - Display username by getting the userId from UserAuthService

8.0 TYUC001 & TYUC002 – View Menu Item List (Admin & Customer)

After successful login, the details and action button on each menu item has to change based on the role. But currently role information is not available. This information is has to be included in the JWT token that is returned by the server. The angular application has to decode the token and get the role.

8.1 Rest API

Authentication Service

- authenticate()
  - Get the role from the security context using the below code
    ```
    String role = SecurityContextHolder.getContext().getAuthentication()
                    .getAuthorities().toArray()[0].toString();
    ```
  - Add role as part of the JSON response by adding a new item into the map.
    ```
    jwt.put("role", role);
    ```

com.cognizant.truyum.SecurityConfig

For accessing the role of a specific user from in memory we need the reference of the

InMemoryUserDetailsManager, as we have not currently defined it as a bean, following changes needs to be done in SecurityConfig.

- public InMemoryUserDetailsManager inMemoryUserDetailsManager()
  - Define @Bean annotation for this method
  - Implement below specified code and return the in memory details manager:

```
@Bean
public InMemoryUserDetailsManager inMemoryUserDetailsManager() {
    LOGGER.info("Start");
    List<UserDetails> userDetailsList = new ArrayList<>();

    userDetailsList.add(
        User.withUsername("user")
            .password(passwordEncoder()
            .encode("pwd"))
            .roles("USER").build());

    userDetailsList.add(
        User.withUsername("admin")
            .password(passwordEncoder()
            .encode("pwd"))
            .roles("ADMIN").build());

    LOGGER.info("End");
    return new InMemoryUserDetailsManager(userDetailsList);
}
```

- configure(AuthenticationManagerBuilder auth)
  - Replace the current code in this method with the code specified below:
    `auth.userDetailsService(inMemoryUserDetailsManager());`

com.cognizant.truyum.controller.MenuItemController

Autowire InMemoryUserDetailsManager. This will be used to get the role details.

- getAllMenuItems()
  - Use code below to get the role of the user

```
Authentication authentication =
      SecurityContextHolder.getContext().getAuthentication();
String user = authentication.getName();
UserDetails userDetails = inMemoryUserDetailsManager.loadUserByUsername(user);
String role = userDetails.getAuthorities().toArray()[0].toString();
```

  - Based on the role either call getMenuItemListAdmin() or getMenuItemListCustomer() from MenuItemService

8.2 Angular

Login Component

- onSubmit()
  - After successful login obtain the role from the response and set the userId and role property in UserAuthService (add additional properties in UserAuthService appropriately).

Menu Items Component

- Include new property role
- constructor
    - Inject UserAuthService
- ngOnInit()
    - Set the role from UserAuthService
- HTML Template
    - Make display of menu item details and action button (“Add to Cart” or “Edit”) based on role

9.0 TYUC003 – Edit Menu Item

9.1 Populate form fields in Edit Menu Item component

9.1.1 Rest API

com.cognizant.truyum.controller.MenuItemController

- Method getMenuItem()
    - @GetMapping(“/{id}”)
    - Autowire MenuItemService
    - Invoke MenuItemService.getMenuItem(id)
    - Return the MenuItem instance

com.cognizant.truyum.controller.MenuItemService

- Method getMenuItem()
    - Autowire MenuItemDao
    - Invoke MenuItemDao.getMenuItem(id)
    - Return the MenuItem reference

com.cognizant.truyum.controller.MenuItemDaoCollectionImpl

Reuse existing code of getMenuItem()

9.1.2 Angular

Menu Item Component

- onEditClick()
    - Route to Edit Menu Item component passing the menuItemId

Menu Item Service

- Get base url from environment
- getMenuItem(id)
    - Invoke rest api ‘/menu-items/[id]’
    - Return menu item details

Edit Menu Item Component

- constructor

    - Inject menu item service
- ngOnInit()
    - Get id from router parameter
    - Invoke getMenuItem(id)
    - Assign the response from getMenuItem(id) to component property that is used to populate the form fields. If it is reactive form create FormGroup and FormControl with appropriate values.

9.2 Save Menu Item

9.2.1 Rest API

com.cognizant.truyum.dao.MenuItemDaoCollectionImpl

Reuse existing code for ModifyMenuItem

com.cognizant.truyum.service.MenuItemService

- Method modifyMenuItem(MenuItem menuItem)
    - Invoke save method in MenuItemDao passing the menuItem

com.cognizant.truyum.controller.MenuItemController

- Method modifyMenuItem(@RequestBody MenuItem menuItem)
    - @PutMapping()
    - Invoke save() method in menu item service passing the menuItem

9.2.2 Angular

Menu Item Service

- save(menuItem)
    - Invoke rest api '/menu-items' with put option and inclusion of form field details in the request body

Edit Menu Item Component

- Create new property named as 'saved' with type boolean. Default value false.
- onSaveClick()
    - Invoke the save() in menu item service passing the menu Item details modified by the user
    - Change boolean value 'saved' to true.

Edit Menu Item HTML Template

- Based on value of 'saved' show/hide the form field section or the success message section.

10.0 TYUC004 – Add Item to Cart

10.1 Rest API

com.cognizant.truyum.dao.CartDao

- Copy and reuse the interface defined in Java Module

com.cognizant.truyum.dao.CartDaoCollectionImpl

- Copy and reuse the class from Java Module.
- Modify the map to type <String, long> instead of <long, long>. This is to handle storing the user as string directly. Modify code to handle other compilation errors on making this change.

com.cognizant.truyum.service.CartService

- Autowire CartDao with CartDaoCollectionImpl
- Include method addCartItem() with parameters the userId and menuItemId
- Invoke addCartItem() in CartDao passing userId and menuItemId

com.cognizant.truyum.controller.CartController

- @RequestMapping("/carts")
- Autowire CartService
- addCartItem() with annotation @PostMapping("/{userId}/{menuItemId}")
  - Get the userId and menuItemId from the parameter
  - Invoke addCartItem() in CartService passing the userId and menuItemId

10.2 Angular

Menu Item Service

- Get base url from environment
- addCartItem()
  - Invoke http call with post mapping for "/carts/" + userId + "/" + menuItemId

Menu Item Component

- onAddToCartClick()
  - Get the menu item id from the component property
  - Get the user id from UserAuthService
  - Invoke addCartItem() method in service passing userId and menuItemId
  - Set respective component property to display the successfully addition of item to cart

11.0 TYUC005 – View Cart Items

11.1 Rest API

com.cognizant.truyum.dao.CartDaoCollectionImpl

The getAllCartItems() method will be reused here.

com.cognizant.truyum.service.CartService

- getAllCartItems()
  - Invoke getAllCartItems() in cartDao

com.cognizant.truyum.controller.CartController

- Autowire CartService
- getAllCartItems()
    - @GetMapping("/{userId}")
    - Invoke getAllCartItems() in CartService passing the userId

11.2 Angular

Cart Service

- Get base url from environment
- Inject UserAuthService
- getAllCartItems()
    - Get userId from UserAuthService
    - Invoke http GET method with url "/carts/" + userId
    - Get the Cart data and return the same

Cart Component

- constructor
    - Inject CartService and UserAuthService
- ngOnInit()
    - Get the userId from
    - Invoke getAllCartItems() and set the cart object in the component which should be able to display the cart items and the total.

12.0 TYUC006 – Remote Cart Item

12.1 Rest API

com.cognizant.truyum.dao.CartDaoCollectionImpl

The deleteCartItems() method will be reused here.

com.cognizant.truyum.service.CartService

- deleteCartItem()
    - Invoke deleteCartItem() in cartDao

com.cognizant.truyum.controller.CartController

- Autowire CartService
- deleteCartItems()
    - @DeleteMapping("/{userId}/{menuItemId}")
    - Invoke deleteCartItem() in CartService passing the userId and menuItemId

12.2 Angular

Cart Service

- Get base url from environment

- Inject UserAuthService
- deleteCartItem()
  - Get userId from UserAuthService
  - Invoke http GET method with url “/carts/” + userId + “/” + menuItemId

Cart Component

- onDeleteClick()
  - Get the userId from UserAuthService
  - Get the menuId from the event
  - Invoke the deleteCartItems() method in cart service passing the userId and menuItemId
  - Display message in html template that cart deletion is successful.

13.0 TYUS006 – Signup

13.1 Rest API

com.cognizant.truyum.model.User

Create attributes that matches with the signup form and create necessary setters and getters.

Include annotations against each attribute to validate the value.

com.cognizant.truyum.controller.UserController

- Define @RequestMapping(“/users”) at class level
- signup(@RequestBody @Valid User user)
  - @PostMapping
  - Using inMemoryUserDetailsManager, check if user exists.
  - If user exists, throw new UserAlreadyExistsException, which should translate into bad request with message “User already exists”
  - If user does not exists add new user with role as “USER”

Refer InMemoryUserDetailsManager API documentation to check user existence and add adding a new user.

13.2 Angular

User

Add user interface representing the form fields in sign up form.

User Service

- Get base url from environment
- addUser(User user)
  - Invoke http POST method with passing form details in request body for service “/users”

User Component

- addUser()
  - Get the form details and invoke
  - Display message in html template that cart deletion is successful.
  - Handle bad request error and display appropriate message. This should display form field validation errors and User already exists message.

14.0 TYUS008 – Logout

14.1 Rest API

No coding required for handling logout.

14.2 Angular

Invoke the logout method in AuthenticationService.

Reset the userId and role in UserAuthService to null.

Navigation Component

Based on AuthenticationService and UserAuthService details the navigation component should get changed accordingly. Test if it works fine.

15.0 Performance Testing

Using JMeter, performance test any specific web service. Identify the number of concurrent requests that can be handled by the server with the threshold response time of 3 seconds.

16.0 Mock MVC Testing

Implement Mock MVC Testing for all possible scenarios of signup rest api.

17.0 Coding Standards and Guidelines

- Never use System.out.println(), use logger.debug() instead
- Include start and end logs in each method
- Include debug logs for data retrieval and flow
- Each Rest Controller class should have RequestMapping that maps to the particular domain entity
- The URL definition should be in all lower case with words separated with hyphen.
- Method level URL definitions in rest controller should contain only the parameters and should not contain any other URL definition.

- Apply the right http method based on the operation performed. POST for creation, PUT for updation, GET for reading data and DELETE for removing data.
- Do not include any instances level variables in controller apart from service auto wiring. Remember that in production environment based on volume of usage, there is a good possibility a rest controller method might be invoked in parallel, having instance variables with user specific information will overwrite one over the over and might result in bad user experience.

18.0 Submission

18.1 Code submission instructions

Use git add, commit and push commands to upload your code into remote GltLab repository.

During commit, give the commit message as “rest-api”.

19.0 Change Log

	Changes Made			
V1.0.0	Initial baseline created on <dd-Mon-yy> by <Name of Author>			
Vx.y.z	<Please refer the configuration control tool / change item status form if the details of changes are maintained separately. If not, the template given below needs to be followed>			
	Section No.	**Changed By**	**Effective Date**	**Changes Effected**