TRAINING GUIDE

Testing the Backend Server

Smart Vendor & Procurement System
redSling Backend

Start the generated backend, review its available routes, and test API requests directly in redSling. Use Test Server Control, the Runtime section, the Route Catalog, and the Request Builder to inspect backend behavior.

1 Open the Backend section

Open the **Backend** section of the **SmartVendorWeb** project. Locate **Test Server Control** to manage the generated test server.

2 Configure the test database

Enter the database connection required for testing in **Custom DB URL**. The example connection is:

```
mongodb://localhost:27017/my-testing-db
```

Replace this example with your test database URL. Leave the field empty to use the default test connection from the main server's **MONGO_URI**.

3 Start and review the runtime

Start the generated backend after the database configuration is ready. In the **Runtime** section, review these details:

Property	Setting
Runtime URL	Address of the running backend
Internal Port	Port used by the runtime
Database Connection	Database connection used for testing
Routes loaded	Number of routes available in the runtime

Refresh the available routes when needed. Use the runtime controls to stop the server when testing is complete.

4 Review the Route Catalog

Review each route's **HTTP method**, **route path**, and any **protected parameters** that must be supplied. Select a route to open it in the **Request Builder**.

Configure and Test API Requests

5 Configure the request

In the **Request Builder**, select the HTTP method and review the route. Supply the required path parameters, query parameters, request headers, and request body.

For authenticated APIs, provide an **Authorization** header with a **Bearer token**. Retrieve the required token from the **Data Models** section. Where required, configure the request body in JSON format.

6 Test the Login API

Select the generated Login API route with the **POST** method:

`/customroutes/custom_model_login/:projectid`

The Request Builder automatically resolves the project ID into the final API URL. Configure the required request values and send the request to the running backend.

7 Test a data route

Select the **table_data** endpoint and provide the required table name as a path parameter. Send the request and inspect the response status and returned data or error message.

In the demonstrated example, the API returns **400** with the message **User not found**. Use the response to identify and troubleshoot the request or backend behavior.

8 Test a POST request

Select the relevant POST route and configure the JSON body with the required user information, such as **name**, **email**, **password**, and **user role**. Send the request and check whether the response confirms that the record was added to the selected table.

9 Review the response

Review the HTTP status and response body after each request. The successful request shown in the demonstration returns **200** with the backend response. Confirm the returned result for the route being tested.

Final review

1. Confirm the intended test database connection.
2. Check the Runtime URL and loaded routes.
3. Verify the method, route, parameters, headers, and body.
4. Send the request and inspect the status and response.
5. Stop the runtime when testing is complete.