CS 8711 Project 3 Json Parsing Summary

Bhagirath Tallapragada and Varchaleswari Ganugapati

Abstract— **In this project, we developed a Python script that processes an input JSON file containing information about data input forms and generates five output files named the same as the "name" field of the data input form. For example, for form2.json, following files are generated: 1) interests.html 2) interests.js 3) interests.sql 4) interests.py 5) interests_display.html. We explain the functionality of each component of this project in this report.**

I. INTRODUCTION

The Python program takes as command line input a JSON file and produces as output the following five files: 1) An HTML file that contains code to display the data input form in a browser. 2) A Javascript file that performs basic validation such as data type checking for text box input as well as check for required fields. In addition, a Javascript function, called submitData, that sends the form data to a backend service for storing in a MySQL database. 3)A MySQL script file that contains create table statements to create relational table(s) that can accept data submitted from the data input form. 4)A backend Python RESTful API program with two endpoints namely: "/webforms/insert" - This endpoint processes data submitted by the user using the data input form. The endpoint should simply insert the data into the database table(s). "/webforms/display"-This endpoint simply returns the contents of the entire database that includes all the table names and table data. 5)An HTML file that contains a button; upon click the page will display the contents of the database (basically a dump of all the tables in the database). The Javascript function, called displayData, reacts to the button click and calls the backend via RESTful call to retrieve the data.

II. COMPONENTS AND CONTRIBUTION

The main component of this project is webForms.py which is a python script that generates the required output files. The following components were written into webForms.py by the project members:

A. Html form by Varchaleswari Ganugapati

Html5 form techniques is used to build the form by using various input tags as form elements. The form action is defined in the javascript file explained below and likewise, the action attribute in the form tag contains a call to validateForm() function.

B. Javascript file by Bhagirath Tallapragada

A Javascript function named validateData() is developed. This performs basic validation checks such as data type checking for text box input as well as check for required fields is added.The form raises alert(s) if the entered data is not in the required format.

A Javascript function, called submitData() is developed. This function converts the form inputs into json format and sends the data in an acceptable form to the backend service that stores the data in a MySQL database.

In addition to this a function named displayData() is also included in this file which is described in the sections below.

C. MySQL script by Bhagirath Tallapragada

The mysql script contains a series of create table statements created based on the requirement of the input json file. The tables created preserve the relation between data through primary and foreign key constraints. In addition to creation the mysql script also ensures deletion of any existing tables by the same name in the target database.

D. Backend python file by Varchaleswari Ganugapati

This python file serves as a middleware to connect the frontend to the backend database. The following two end points are used: 1)"/webforms/insert/" - This end point receives data from in form of json from the front end user interface and inserts the data into relevant tables in the backend. There could potentially be more than one insert statement depending on the kind of input from the form. For example: If the input json file has one checkbox and one multiselect list along with other inputs such as textbox or radiobuttons, maximum number of tables would be three, one for the main table holding textboxes and radiobuttons, and the other two for the auxiliary tables of checkbox or multiselectlist. Similarly the number of tables will follow the same pattern depending on the number of auxillary tables needed. 2)"/webforms/display/" - This end point recieves the json data such as database name, userId/ password and extracts all the tables in the database by querying "show databases". It then uses the output from executed query to obtain all the table names and selects all the content from the table. The result is sent to the upstream in json format.

E. Display Data by Bhagirath and Varchaleswari

This piece of work was divided into html page designing and coding the javascript function "displayData". The html page is built by Varchaleswari and Javascript function by Bhagirath. Html page contains a button, which upon clicking will generate a tabular form of all the contents in the backend tables. Javascript function does a RESTful call to the end point and obtains the entire data in json format. This data is then formatted into table and then rendered as html.

F. Json Schema Validation by Bhagirath and Varchaleswari

We also created a json schema to validate the input json file that is being sent in the command line. Schema validates for the file structure i.e. if the set of elements are present in the required format. Few attributes such as ename, etype and caption should be present. "Group" sub element of the elements tag should be in the required format and contain caption and value attribute values. "validateJson" function in the webforms.py file does the required validation and prints on the terminal if the validation was successful or not.

III. A VIEW TO THE EXECUTION FLOW AND SAMPLES

The following is a step-by-step description of the execution flow:

1)A form.json data is passed in the command line as input parameter to the WebForms.py file.

2)Using the data in the input json file corresponding html files and javascript files are created along with a backend python file.

3)The html page is launched and data is entered in the form using the input fields available in the UI. Then, the submit button is clicked.

4)On form submission by clicking submit button, the javascript invokes the listener function to the form action event and first validates the data in the input form. After validation, the data is sent to the backend via an ajax RESTful api call to the given endpoint. If the data is not passed in requisite format, alert(s) are generated. 5)The API expects the data in a format as shown in fig 1. The API resolves the call and checks if the input had any checkboxes or multiselectlists or not based on which the insert statments are created. For example if a input form has one checkbox and one multiselectlist then three insert statements will be created one for the main table, one for the checkbox data and one for the multiselectlist data. On successful insertion the API returns result as follows:

{"ok": True, "message": "Record successfully added!"}

Further, this format is used by the javascript function to display the message on the form page.

6)For displaying the content from the database another webpage is created which will contain a button saying "display". Onclick of this button will invoke the listener in the javascript which will call "/webforms/display" along with passing the relevant data as shown in fig 2. The API will first invoke "show tables" for the given database and then will do a "SELECT * from <table_name>" for each table in the database. The result will be stored in a dictionary and will be sent in the success callback in a json format as shown in fig 3.

IV. CONCLUSIONS

Through this project we implemented a python program that parses an input json file to generate the required files for a web application that performs form validation and submission to a backend database. All use cases were tested thoroughly to ensure error free generation of the model's workflow.

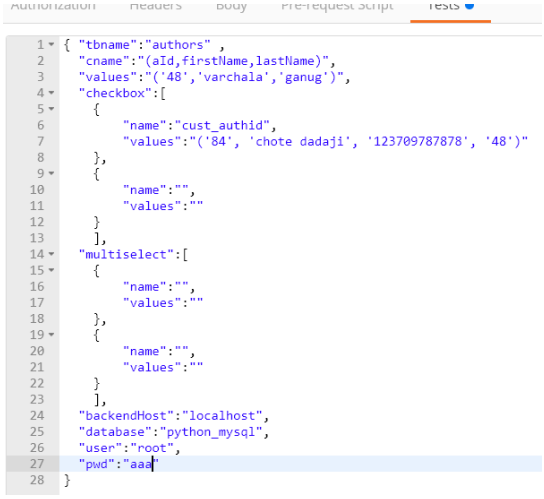


Fig. 1: Json input format for /webforms/insert call

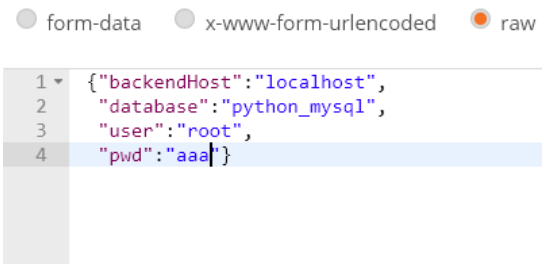
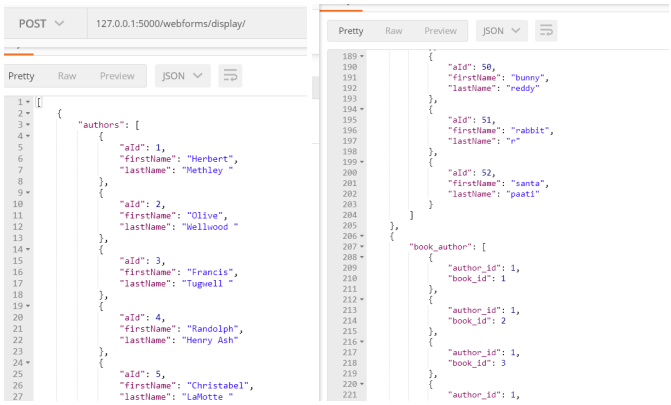


Fig. 2: Json input format for /webforms/display call

REFERENCES

[1] http://tinman.cs.gsu.edu/ raj/8711/sp21/p3/



(a) Json output (b) Json output extended

Fig. 3: Json output from /webforms/display call