Summer 2018

Summer student report

Starting geant-val V2 development

Luc Freyermuth
Supervisors: Dmitri Konstantinov & Gabriele Cosmo

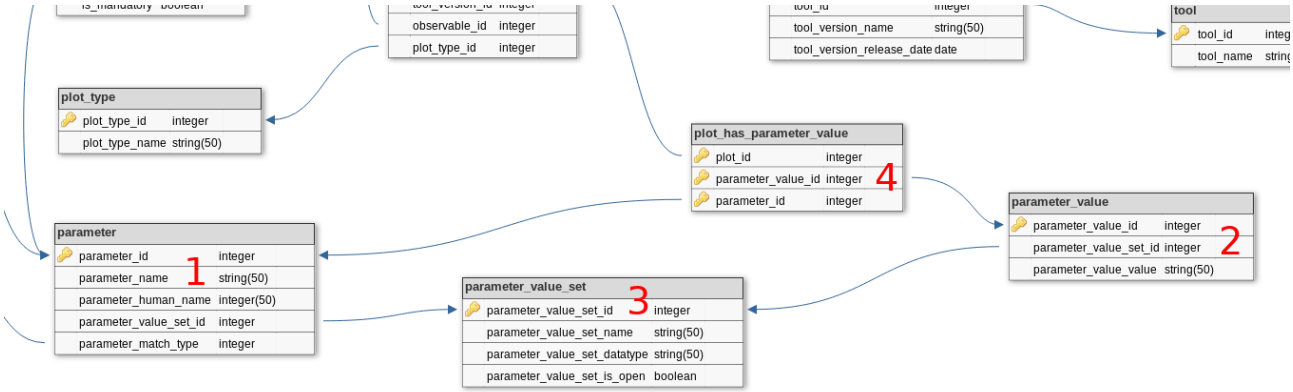
Table of contents

1. Introduction

Geant-val is a tool for Geant4 regression testing and validation. This web application has already been in production for several years and I should have been working on it this Summer.
However, the actual geant-val suffers from its historical data structure: the project was originally built to handle fixed target test with a limited number of parameters only. A lot of other tests have been implemented since, but ingenious workarounds are usually needed, requiring a lot of work and time.
Moreover, the technology used for the front end, AngularJS, has entered its Long Term Support phase and wont be supported anymore in a few years.
So, after several discussions with my supervisor, we decided that I would work on a new version of geant-val that would solve these problems directly at the root.

2. Data structure

Before, the list of parameters that a test is using was hard coded in the database structure. This was the main reason of most of the problems, so I proposed that the list of parameters and their values could be instead listed in dedicated database tables. Then we built the whole database schema around these tables.



Part of the data schema representing test parameters

(1) The *parameter* table that lists all the parameters handled by geant-val V2. This makes easy to create new parameters by creating a new entry in this table.
(2) The *parameter_value* table stores all the possible values of the parameters.
(3) The *parameter_value_set* table links parameters to their values. This intermediate tables allows for several parameters to share the same possible values (Example: the *beam particle* and *secondary particle* parameters share the *particle* value set)
(4) A linking table table is used to indicate which data uses which parameter with which value.

In general, in the whole data schema, we tried to avoid hard coding as much information as possible, so that future features can be implemented without having to change the schema as much as possible.

3. Geant-val application

After changing the data structure in depth, the whole logic of the server had to be changed. It was a good time to think again about the technologies we are using for the application in general.
For the server, I proposed to use, in addition to NodeJS and Express, a framework to create basic REST API endpoints easily and automatically.
For the client, we had to move from AngularJS that entered its Long Term Support stage.

3.1 Server

After testing different available solutions, we decided to use LoopBack to handle our REST API. LoopBack is able to create our database table and generate REST endpoints based on our data structure (models). This solution has several advantages:

- The API will be more secured (the framework is maintained by a team who fixes security breachs found by the community).
- The API will follow good practices (route names and methods will be normalized).
- It lowers the amount of required work.
- We can still implement our own logic for more advanced use cases.

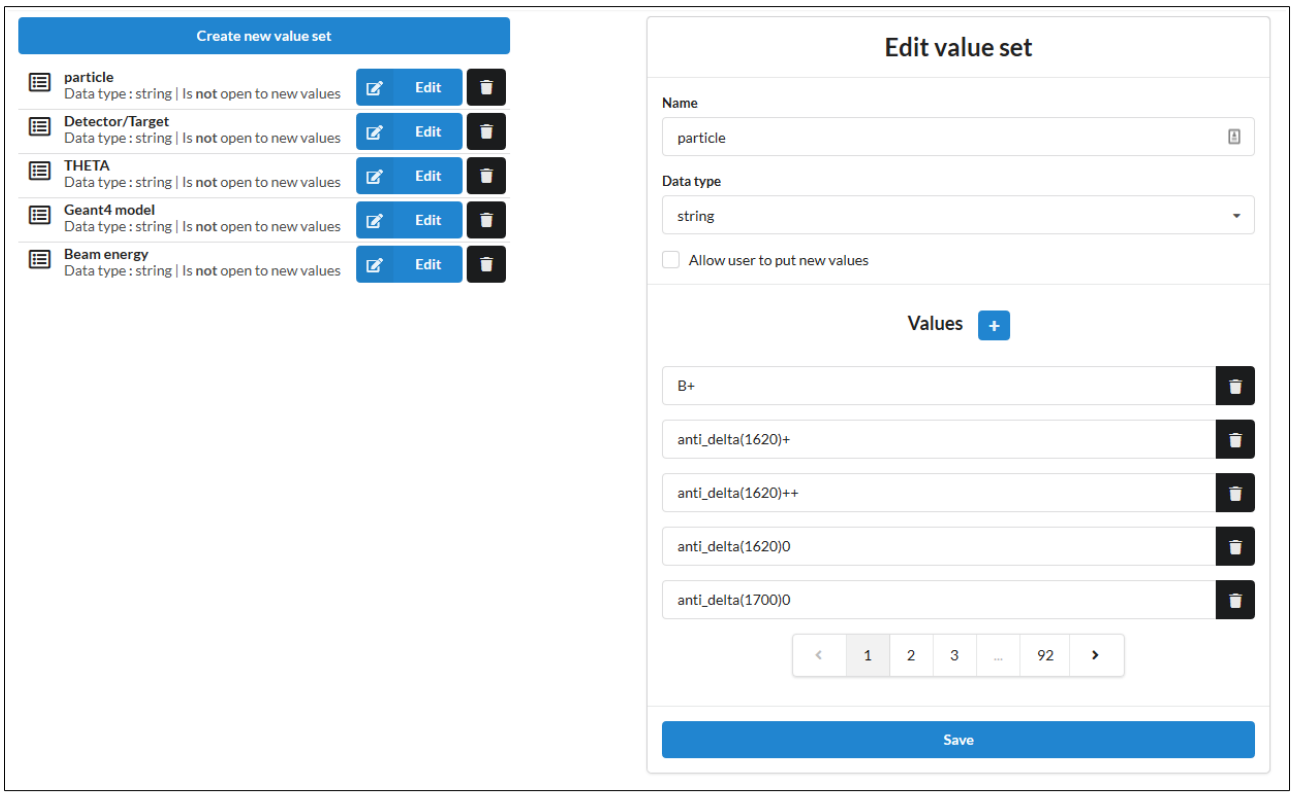
After this, I implemented our data schema as LoopBack models. Then I mainly moved to work on the front end, but I still had to come back to write some logic, especially advanced SQL queries that LoopBack couldn’t handle itself. My colleagues worked on custom API endpoints and managing access rights.

3.2 Client

In order to replace the aging AngularJS, I proposed to move to Angular. Angular could be thought as a new version of AngularJS, but it is completely different and just inherited of the name of the previous JavaScript framework of Google.
Angular natively uses TypeScript, a superset of JavaScript that adds optional typing and makes easier to scale application. It relies a lot on web components, encapsulated and reusable parts of web pages.

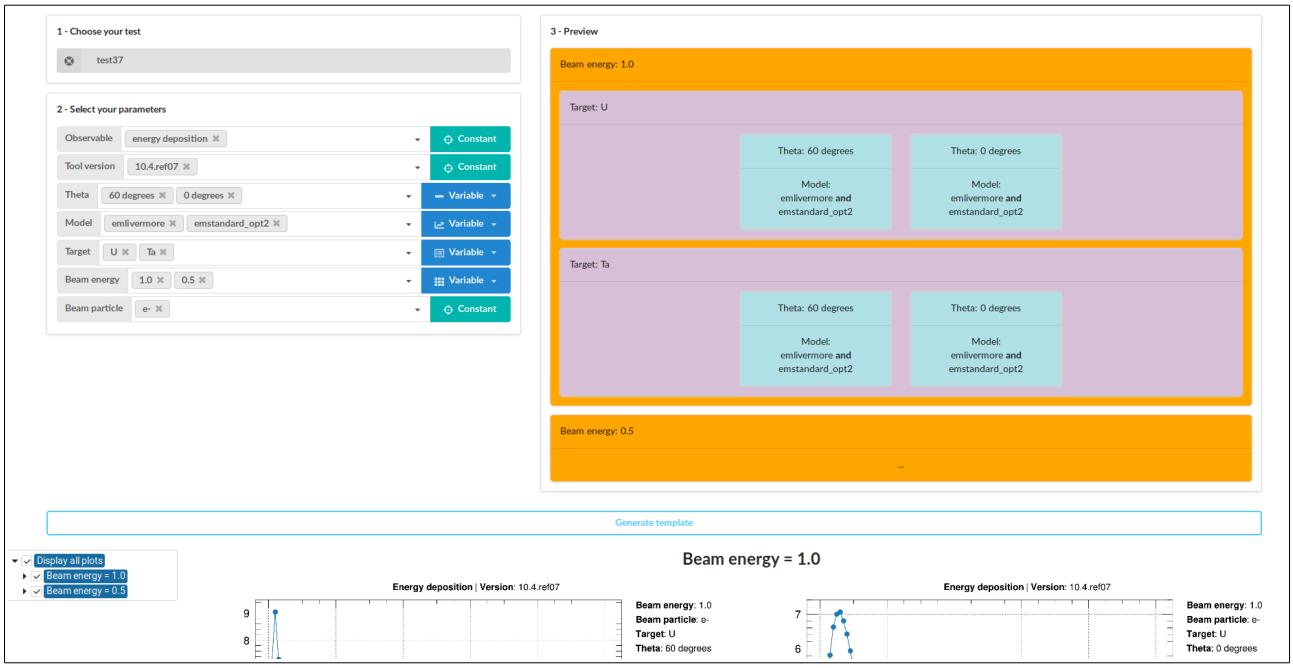
I started by building the core of the app by implementing our data models and the main shared services such as the authentication service, or the API services that will be used everywhere in the app.

Then, we built the administration of the website, to be able to insert our old data in the new database schema and test it.



Admin panel to manage parameter value sets

Once we had data in our new database, I created a first user page that could use the new schema in an interesting way. It is a sort of “proof of concept”. This page allows users to create a template (with real time preview) to display ordered plots and compare data.



The template generator

4. Conclusion

Restarting a project from scratch takes a lot of time and the second version of geant-val is obviously not finished yet. However, I was able to participate a lot and tried to put the project on a good way. It’s impossible to anticipate all the future problems and ideas, so I tried to create a data schema as flexible as possible to allow future ideas to be implemented. In the same way, the client is built to be as modular as possible, so most of the features can be moved or removed if necessary, without breaking the app. It was a pleasure for me to work in an environment where my ideas were taken into account and where criticism was always constructive.