Web API for Sasview

SHIP Student: Ting (Xael) Shan   Mentors: Jeff Krzywon, Elizabeth Kelley

CHRNS — CENTER FOR HIGH RESOLUTION NEUTRON SCATTERING

NIST CENTER FOR NEUTRON RESEARCH

WHAT IS SASVIEW?

NIST Center of Neutron Research is a user facility that uses neutron scattering to observe particle elements. One subdivision is Small Angle Neutron Scattering (SANS):

- Instrument shoots neutron through a sample
- Neutron scatters after collision with sample particles and hit detector
- Scattering pattern, scattering length density, intensity, and more elements are measured

Sasview is analysis software that takes the data collected and analyzes it into derivable data.

Two category of tools:
- Analysis Tools: Fit, Inversion, Invarient, Corfunc
- Calculation Tools

WHAT IS AN API

An API is an application programming interface. It acts as a "translator" so the user can "communicate" with the server. The user and server do not send the same data format, so translation is necessary.

Petfinder is an example of a user-friendly API:

Currently, Sasview has a Graphic User Interface (GUI) — allows user to interact using icons, buttons, and text.
The API act parallel to GUI as second way to communicate to the server.

BENEFITS OF AN API FOR SASVIEW

Alleviate Resources

Analysis tools can take weeks with too much data.
Calculations are done remotely, does not use user's GPU/CPU

Experimental benefits

Automation!
The API allows experimenters to continuously ask the server to update their analysis with code.

Operating System Deployment

Overcomes Window, Linux, Mac Requirements as the computation is being done elsewhere.

DATA → SCRIPT → New request → RESTFUL API (TRANSLATES) → Sasview Server (COMPUTATION)

DEVELOPING AN API FROM SCRATCH

Web Framework: Django
- Modular layout   - Pre-included features and tools   - Easy to develop frontend

Modularity: Set up for future success. Allow future developers to pick up developing this API easier

Only Backend: the part that is responsible for storing and manipulating data

Main files
NEW APP
models.py -> databases
views.py -> functions
URL linking

- Links everything together!
- Allows the user to get to all functions

Base URLs -> path(""), re_path("data/"), re_path("analyze/")
Spec URLs -> API functions
path("login/"), path("get token/"), path("upload/"), path("download/"), path("info/"), path("start/"), path("models/"), path("optimizers/")
path("fit/") → path("fit/"), path("models/"), path("optimizers/")

- Database structure
- Define what the user inputs
- Make sure user can't put in malicious data

SQLite Database format is like a table:

	Data id	user	file	file_name	is_public
Field: we define					
Data: user inputs	1	None	<File_obj>	Data.txt	True
	2	None			
	3…	None			

- Translator
- Puts into databases
- Defines how user's data looks

JSON format:
"is_public" : False
"file":file_obj
"file_name" : "hello.txt"

Data id	user	file	file_name	is_public
1	None	<File_obj>	Data.txt	True
2	None	<File_obj>	Hello.txt	False
3…	None			

Inputted data

URL data: 127.0.0.1:8000/v1/data/upload/?data_id=1

USER SIDE

There are multiple ways to be a user. The primarily ones are:

Development Server: API view   Terminal/Command pallet   Script

This demonstration of the working model is using the development server, as it is the most readable

GET – Read Data   URL 127.0.0.1:8000/v1/data/list/   Response:
No Inputted Data

POST – Insert Data   URL 127.0.0.1:8000/v1/analyze/fit/

Inputted Data:
```
{
    "model":"cylinder",
    "data_id":22,
    "optimizer":"amoeba",
    "parameters" :
    [
        {
            "name":"radius",
            "value":35,
            "data_type":"int",
            "lower_limit":1,
            "upper_limit":50,
            "analyze":true,
        },
        {
            "name":"length",
            "value":350,
            "data_type":"int",
            "lower_limit":1,
            "upper_limit":500,
            "analyze":true,
        }......}
```

Response:

FUTURE

WHAT WILL HAPPEN TO THIS AFTER THIS SUMMER?

This will continue to live as an in-progress project for Remote fitting interface in Sasview GitHub

NEXT STEPS:
- Add constraints to Fit   - Implement calculation tools
- Implement other analysis tools   - Wrap and create frontend

WHAT WILL I CONTRIBUTE AFTER?
- Use API to analyze bicelle data
- Get API approved to merge into Sasview package