Development of Web Tools for LTX-beta

Nathaniel Sokolow (Community College of Baltimore County), advised by Robert Kaita (PPPL)

the Lithium Tokkamak eXperiment (LTX-beta)

- LTX is a small tokamak which is designed to experiment with liquid lithium walls.
- Lithium is used in tokamaks to reduce plasma recycling. Plasma recycling occurs when hot plasma ions touch the walls, cool off, and re-enter the plasma, which causes edge cooling and energy loss.
- The lithium is evaporated to coat the quadrants of stainless steel shell, which form the plasma facing components (**pfc**).

Figure 1: CAD model of LTX. The lithium is evaporated onto the inner shell. As shown, that shell consists of four quadrants – it is split vertically and horizonally through the center. Those quadrants are then vacuum sealed into a single shell

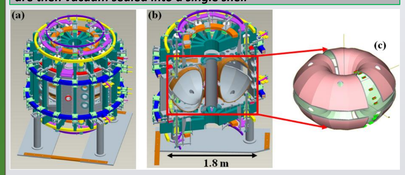
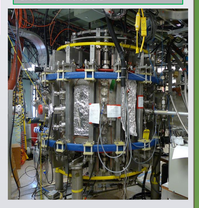


Figure 2: Photo of the LTX external facade

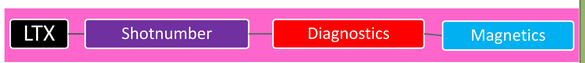


LTX-Beta Data Acquisition

Overview

- LTX takes several dozen plasma **shots** per day when running. A **shot** involves running the machine and creating a new plasma.
- Every shot, various diagnostics are used to collect different plasma parameters, which are necessary to understand the results of the experiment. Some of these are direct measurements, such as plasma current, and others are derived from the known quantities, such as electron density and temperature.
- After the diagnostics collect the data, that data is digitized if necessary, and then stored as a signal built from the numbers, in a database. That database is called MDSplus.
- **MDSplus** is a ***tree*** database- meaning that there are levels. The bottom level with the actual data is called a ***node***.
- Figure 3 is a good example. The top level is LTX. Inside LTX are the trees for each shot (shotnumber). Inside each shot are different categories of diagnostics, in this case magnetics. Inside magnetics are the different kinds of magnetic diagnostics, like Mirnov Coils (used to measure magnetic field strength at a point). Inside the Mirnov diagnostics is the Mirnov we are looking for, which itself has several attributes. The picture shows the first part of this.
- This is similar to a computer directory [aka Folder] on a computer. The pathname to raw data from a Mirnov Coil would be: ***LTX/shotnum/Diagnostics/Magnetics/Mirnovs/#18/Raw_Data***

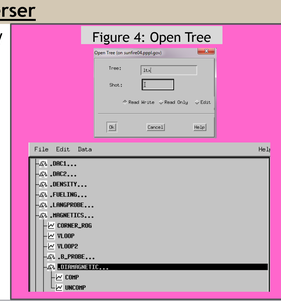
Figure 3 – First Part of an MDS path



Traverser

- Traverser is a built-in MDSplus utility to used to access the LTX data tree.
- Traverser is great for analysis or changing of the tree, but it is not meant to access the data itself.
- It prompts the user for a **tree name** and **shotnumber**, and sends the user to that subtree.
- Traverser can be used to add or edit nodes as well. The tree can be opened in read-only, r/w, or edit mode.

Figure 4: Open Tree

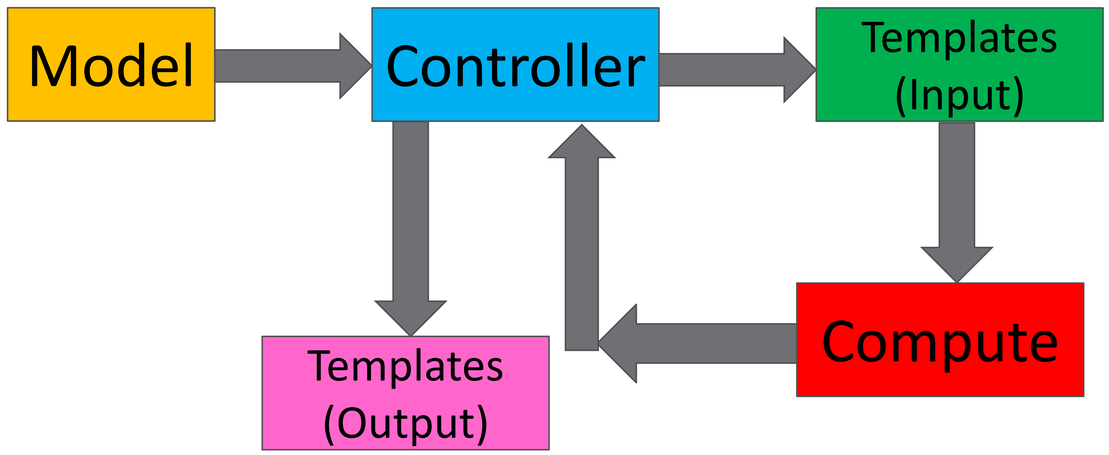


LTX Web Tools

This summer, development commenced on a suite of web tools (basically web applications) which could be used to plot MDSplus data. Hopefully in the future the scope of the project will expand beyond just plotting, and may eventually be used on NSTX.

See the right panel first for an overview of LTX plotting through MDS and Scripting

Data Structure Diagram - How a Flask Application Works

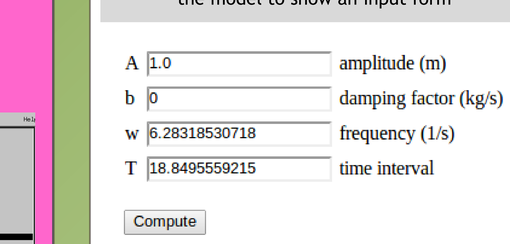


Definitions

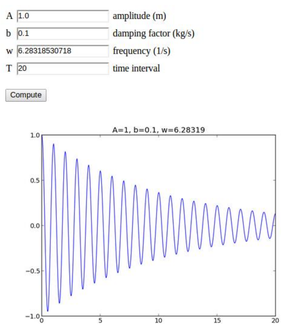
- **Model**: The source for the input page. Written as a Python class, in this application it provides the HTML input form in which the user inputs the shotnumber and node they want. The input template renders the model as HTML for your browser to show when you first go to the site.
- *The main advantage of writing this in Python instead of manually making the HTML form is the convenience of having certain commands built into Python already, most notably form validation (making sure the inputs fit the format expected – no HTML commands etc.)*
- **Controller**: The part of the Application that actually contains the commands to show the input template (the model); and calls the compute method. It then saves the plot output from the compute method, and adds that to the **output template**. This is the app the server runs.
- **Compute**: Accepts the user parameters input to it by the Controller method (from the input form), and uses them to output a plot. For LTX, this is a script which connects to the LTX tree and plots the data from the desired node to the template.

The photos below are from a different project, but the projects are conceptually the same.

Figure 4: The Input Template. The controller instructs the HTML to use the form class from the model to show an input form



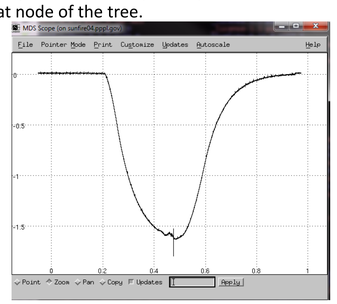
- When the Compute button is clicked, the HTML input form is submitted.
- A request is sent to the handler (the Controller), using the parameters in the fields of the form
- The controller runs the compute function with those parameters.
- The compute function contains the equation, with A=1.0, b=0, etc., and plots the function using that
- The compute function returns the plot to the controller.
- The controller adds the plot to the template, which outputs the figure at the right.



Plotting LTX Data

Plotting MDSplus Data Through Scope

- MDSplus has a built in plotting utility, called mdscope. This can be used from any MDSplus installation. It is a GUI for MDSplus data plotting. There is a newer, Java based version called Jscope.
- It asks for an experiment name (i.e. LTX) and a shotnumber, and plots the data from that node of the tree.



MDSplus Plotting And Other Uses With Scripts

- The best way to access **MDSplus** is through programming scripts, using them to plot, save, or print the data from a node.
- Most popular programming languages can be used – **Matlab**, and **IDL** are two popular ones. **Object Oriented Languages** like **Python, Java, and C** are useful. **MDSplus** has an object based interface.
- At PPPL, the most common languages are **IDL, MATLAB, and Python**. **Python** is considered the best language to learn now.
- A **Python** example of an MDSplus script is pictured here.

```
#!/p/fdp/anaconda/bin/python #path to Python
from MDSplus import *          # allows MDS functionality
import matplotlib.pyplot as plt     #plotting functions

Connection("lithos.pppl.gov:8000") #connects to the server hosting tree
t = Tree('ltx', 1504291255)  #opens ltx tree with a given shotnumber
ip = t.getNode("Langprobe:I_SINGLE")  #accesses node
data = ip.record.data()          #accesses array of data inside node
time = ip.dim_of().data()   #gets timescale of data inside node
y = plt.plot(timescale, dat.record.data()) #plots data vs. timescale
#parametrizing it as y allows for easier production of labels, legends etc.
plt.show() #outputs all plots
```

- First, it connects to the server port the tree is hosted at with MDSconnect This function can even be used remotely if one is inside the firewall.
- Then it uses the getNode command to access the data array it will plot
- After that, it uses normal Python Matplotlib functions to plot and save a figure.
- Other uses of MDS scripts can include: Listing the data stored in a node into a CSV (Comma Seperated Variable) file, which can be imported into an Excel spreadsheet; creating a logbook from metadata stored in the tree, and watching videos taken by the fast camera of the plasma.

CONTACTS

NSTX Web Tools site: http://nstx.pppl.gov/nstx/Software/WebTools/index.html
Advisor: Bob Kaita - kaita@pppl.gov
Web Tools Build and Scripts: Nathaniel Sokolow - nathanielsokolow@gmail.com
Administrative and Server: Kenny Silber - ksilber@pppl.gov
MDSplus Installation: Greg Tchilinguarian - gtchilin@pppl.gov
LTX Web Tools Site: https://ltx-wtools.pppl.gov/
Davis paper: http://nstx.pppl.gov/nstx/Software/Publications/MDSplus_WebTools_Jan_2002.htm

Acknowledgments

- ***This work was made possible by funding from the Department of Energy for the Summer Undergraduate Laboratory Internship (SULI) program. This work is supported by the US DOE Contract No.DE-AC02-09CH11466.***
- *Bob as my adviser was always willing to sign off on whatever technical flight of fancy I thought necessary for the project, even if I didn't fully understand what I was doing at the time.*
- *Much of the LTX team assisted me in the projects on this poster. Most notably, Paul Hughes, Dennis Boyle, and Drew Elliott were always willing to help when I needed something, and Gus Smalley was always available for technical stuff, from scope readings to wire soldering to anything else.*
- *A special thank you to Greg Tchilinguarian, Danny Ascione, and Kenny Silber, for allowing me to bug them for information about computer issues no one else could have helped with.*