Embedding Interactive Python Web Applications into Electronic Textbooks

Authors:
Sutchenkov Anton, Tikhonov Anton
National Research University "MPEI"

Speacker’s
Anton Sutchenkov
NRU MPEI
**Introduction**

<table>
<thead>
<tr>
<th>MOOC - Massive Open Online Courses</th>
<th>SOOC, SPOC – Small (Open/Private) Online Courses</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Students</strong></td>
<td></td>
</tr>
<tr>
<td>Tens of thousands</td>
<td>Tens or hundreds (but hundreds courses)</td>
</tr>
<tr>
<td><strong>Development and maintenance</strong></td>
<td></td>
</tr>
<tr>
<td>Professional developers, administrative staff; high quality media content; interactive applications; hundreds of thousands of dollars for MOOC development;</td>
<td>The teachers with minimal support from professional developers; Static educational content for SPOC and SOOC may be created with Content Management Systems (CMS) or Learning Management Systems (LMS). <strong>Development of interactive applications embedded in SOOC (SPOC) is a complex and resource-intensive problem</strong></td>
</tr>
</tbody>
</table>
Requirements for interactive web applications for SOOC and SPOC

1. The zero installation of software on student computing devices. Interactive applications should not impose additional requirements for students computing devices.

2. The simplicity and low labor intensity of creating and using interactive applications in the educational process.

3. The automated creation of a graphical user interface based on input and output parameters annotations.
The stages of interactive applications development

1. problem solution;
2. visualization of the results;
3. preparation and publication of the problem and application description;
4. development of the graphical user interface for interaction with the application;
5. application integration into the electronic textbook.
Python ecosystem for interactive applications development

1. The simplicity and the ability to develop applications using a limited subset of language tools and mastering new tools as needed.


3. The Jupyter Notebook (JN), JupyterLab development environments.
The tools for interactive applications development in Phyton

1. **Voilà** - JN extension, supports interactive JN widgets and allows you to easily convert JN notebook into a standalone web application using the JN kernel.

2. **PyViz Panel** - free Python library for creating and using interactive, standalone web applications as well as running them in the JN environment.
The main purpose of the technology is to quickly add the user web-interface, interactive graphics, providing interaction between the server part of the application and presentation the results in the client part of the application. Client part runs in the browser.

Dash applications are fully written in Python. To create a web user interface, there are libraries that provides wrappers for all HTML tags for Python, and allow to create user interface components, including input fields, buttons, check boxes, switches, sliders, etc.

The libraries are extensible and free!
The sequence of user actions when working with an interactive application:

1. **data entry** using the user interface widgets;
2. **transfer control** to the application by clicking the Run button;
3. **application execution** to convert the input parameters values into output parameters values;
4. **visualization** of the obtained results;
5. **interpretation** of the results by the user;
6. **moving to item 1** when performing multivariate calculations.
The automation of interactive applications development

To automate the conversion of the Python application to the Dash or Panel application, the Python application must be designed as a function.

The input parameters displayed in the user interface must be passed through the function header and the function should return the output parameters.

Input and output interface parameters:
1. scalar variables – integers, floating point numbers, logical values;
2. objects including tuples, lists, NumPy arrays, Figure matplotlib objects and Plotly objects
The automation is implemented with a help of @dash_interact() decorator create a web user interface for the function. Three named parameters must be passed to the decorator: inputs, outputs (param_db objects) and layout. Layout parameter contains information about the arrangement of the user interface widgets.

Two modules with Python web application source codes are generated:

- Flask web application – used for debugging user interface and production environment
- module for embedding into Django web applications into an electronic tutorial
**Example of transformation Python code into interactive Web app**

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>import dash_interact import param_db as p</td>
<td>Import <code>dash_interact</code> decorator and <code>param_db</code> library to describe user interface elements</td>
</tr>
<tr>
<td>2</td>
<td>@dash_interact(inputs={</td>
<td>Tree input parameters in the user interface: <code>title</code>, <code>chrgs</code>, <code>pal</code></td>
</tr>
<tr>
<td>3</td>
<td>title:p.String('Electric charges'),</td>
<td>the application header</td>
</tr>
<tr>
<td>4</td>
<td>chrgs:p.NumericTuples(default=((1.,0.1,0.1), (-1.,0.9,0.1),(1.,0.5,0.9)),doc='Charges and coordinates:'),</td>
<td>description of charges</td>
</tr>
<tr>
<td>5</td>
<td>pal:p.ObjectSelector(default = 'binary_r', objects=['binary_r','binary', 'jet', 'Spectral'], doc='Palette:'),</td>
<td>a set of color palettes for visualization</td>
</tr>
<tr>
<td>6</td>
<td>outputs={fig:p.mplFigure()}</td>
<td>output parameter – matplotlib Figure object</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>) # end of @dash_interact</td>
</tr>
<tr>
<td>8</td>
<td>def charges(chrgs, pal):</td>
<td>The function calculates the potential and the electric field strength of the charges</td>
</tr>
<tr>
<td>9</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>return fig</td>
<td>Figure object to visualize the electric field</td>
</tr>
</tbody>
</table>
Visualization of the electric field of charges

Electric charges

Charges and coordinates:

(-1., 0.9, 0.1), (1., 0.5, 0.9)

Palette:

binary_r

Run
Interactive Julia fractal set modelling.

You can select:

- variant
- palette
- area in the picture with the mouse
Conclusion

✓ The tools described in this article allow to easily turn the Python functions into interactive web applications that have a web user interface used both for demonstration purposes in lectures and to solve practical problems.

✓ Interactive applications can be easily deployed and integrated into electronic textbooks, SPOCs, and SOOCs.

✓ The attractiveness of Plotly Dash technology lies in the possibility to create client components to simulate the operation of virtual devices for use in virtual laboratory workshops embedded in electronic textbooks.
Thank you for attention!

Speaker’s contacts:

Anton Sutchenkov
NRU “MPEI”
SutchenkovAA@mpei.ru
http://etm.mpei.ru