Library Hi Tech

Using web2py Python framework for creating data-driven web applications in the academic library

Mathew Miles

Received 12 August 2015
Revised 6 October 2015
Accepted 29 October 2015

Using web2py Python framework for creating data-driven web applications in the academic library

Mathew Miles
Brigham Young University – Idaho, Rexburg, Idaho, USA

Abstract

Purpose – Many libraries have a need to develop their own data-driven web applications, but their technical staff often lacks the required specialized training – which includes knowledge of SQL, a web application language like PHP, JavaScript, CSS, and jQuery. The web2py framework greatly reduces the learning curve for creating data-driven websites by focussing on three main goals: ease of use; rapid development; and security. web2py follows a strict MVC framework where the controls and web templates are all written in pure Python. No additional templating language is required. The paper aims to discuss these issues.

Design/methodology/approach – There are many frameworks available for creating database-driven web applications. The author had used ColdFusion for many years but wanted to move to a more complete web framework which was also open source.

Findings – After evaluating a number of Python frameworks, web2py was found to provide the best combination of functionality and ease of use. This paper focusses on the strengths of web2py and not the specifics of evaluating the different frameworks.

Practical implications – Librarians who feel that they do not have the skills to create data-driven websites in other frameworks might find that they can develop them in web2py. It is a good web application framework to start with, which might also provide a gateway to other frameworks.

Originality/value – web2py is an open source framework that could have great benefit for those who may have struggled to create database-driven websites in other frameworks or languages.

Keywords Digital libraries, Software evaluation, Software tools, Websites, Python frameworks, Web2py

Paper type Technical paper

1. Introduction

I graduated from library school in 1992 with little technical training at all, let alone programming skills. My first job was working for a library software company where I began to acquire technical skills and was introduced to reading code and basic scripting. I left the software company in 1996 and have worked as a systems librarian since that time. With the evolution of the web, I began to teach myself how to create database applications using ASP and later ColdFusion. ColdFusion was a language that I was able to implement with only the documentation and no outside assistance or training, which was important since nobody else on our campus at the time was doing much with creating web applications. It was also a fairly simple language to read and maintain. I would sometimes go months without using ColdFusion at all. When I came back to it, I could remember enough to create new applications and modify old ones. As time went on, I realized that I could not keep up with the pace of change occurring with web programming. ColdFusion continued to evolve, simplifying the process of implementing jQuery widgets for example, but a viable and robust web framework for ColdFusion did not seem to exist. The skills that I had used in the past to create simple database applications were no longer sufficient

Library Hi Tech
Vol. 34 No. 1, 2016
pp. 164-171

0737-8831
DOI 10.1108/LHT-08-2015-0082

to meet the requirements of new applications. In addition, if I wanted to share the applications that I had written in ColdFusion with others, they would also be required to purchase the proprietary ColdFusion server.

One of the biggest obstacles in creating a database-driven web application for the developer is the number of technologies in which proficiency is required – including SQL, JavaScript, CSS, jQuery in addition to an application language such as PHP or Java. Over the past ten years any number of web application frameworks have been developed in a variety of languages as ways to integrate these different technologies together and to reduce the number of mundane tasks associated with creating any web application by "baking in" functionality. Examples include Laravel (PHP), Struts (Java), Ruby on Rails (Ruby), and web2py (Python). I decided to evaluate some of the frameworks written in Python because I already had some familiarity with the language and would not be starting from scratch with a new language. I first tried Django, which is the framework used to develop the popular social media site Pinterest. I found many useful and interesting features, but after spending a number of hours going through the tutorials, I still had not created a simple application beyond the ones I had created from the tutorials. I next tried Flask, which is a much lighter weight framework than Django, but it did not have some of the functionality I was looking for, like built-in authentication and security groups. It also required a fair amount of time to configure an environment where I could start developing. Lastly, I downloaded web2py. Within minutes I had a development environment running on my Mac and PC. Within a couple of days I had written my first project from the tutorial and began writing a new application to port a project from ColdFusion. I have continued using the framework developing a number of small applications including an employee-scheduling tool, a file upload tool, and a URL resolver.

2. Goals of web2py

Massimo de Pierro, the main author of web2py, has written, "I believe that the ability to easily build high quality web applications is of critical importance for the growth of a free and open society. This prevents the biggest players from monopolizing the flow of information" (Di Pierro, 2010/2013). The web2py framework greatly reduces the learning curve for creating data-driven websites by focussing on three main goals: rapid development, ease of use, and security.

2.1 Rapid development and ease of use

The problem with many frameworks is the steep learning curve upon introduction. web2py greatly simplifies this: everything you need to get started is conveniently bundled together. You simply download the web2py distribution, start up the built-in Rocket webserver, and you are ready to go. Though I now use Pycharm as my Integrated Development Environment (IDE), web2py includes an IDE, which is quite serviceable.

web2py is designed with Model-View-Controller (MVC) from the ground up. You do not really have to think about structuring an application in MVC. The framework itself enforces the design.

Since the framework supports SQLite, which is included in the web2py distribution, you can start creating data-driven applications without having to set up and connect to an external database server. The Database Abstraction Layer (DAL) allows you to convert the SQLite application to any of the supported databases that also include PostgresSQL, MySQL, Oracle, MSSQL, Firebird, DB2, Informix, Ingres, Cubrid, Sybase, Teradata, SAPDB, MongoDB, and IMAP. Sqlite3, pymsql, pg8000, and imaplib ship with web2py. The other adapters would need to be installed (Di Pierro, 2010/2013).

Most frameworks require you to learn a different templating language for the views (typically html files). web2py allows you to leverage your expertise that you might already have with Python. The controllers are all written in pure Python, as are the templates. You do not need to learn a separate templating language.

A variety of deployment options are also available for web2py. The easiest of these is to use www.pythonanywhere.com. They have built-in web2py support along with database support for SQLite, MySQL, and PostgresSQL. You can get started by setting up a free account.

2.2 Framework design

The MVC pattern is well supported by web2py. Less experienced developers like myself are familiar with the pattern, but without a framework to enforce the pattern it is difficult in practice to implement. web2py separates the data tables into model files. When you create a data table representation in a model file, the DAL generates the table in the target database. The view or presentation to the user is exposed in html files, and the application logic is written in the controller files. Figure 1 shows the web2py Folder Structure.

2.3 Security

The ability to create and manage user settings for security and personalization was a major requirement for a number of projects I was tasked to create. Previously, user security was one of the biggest headaches because it all had to be written from scratch. web2py has a security scaffolding built into every project you create. User, group, and security role tables are automatically generated and can be easily referenced by other tables in the application.

The DAL allows the user to create schema definitions, which are then translated into database tables in the target database, forming jQuery widgets that are serialized in html templates. The generated forms provide field validation automatically, while the DAL blocks malicious SQL Injection.

3. Examples

I have written three different applications in web2py: an employee-scheduling tool, an obituary index, and a file upload tool. I will show examples from the file upload tool to illustrate some of the core features of web2yp.

3.1 Security tables

When you create a new web2py application it automatically creates a number of tables for you including auth_user, auth_group, and auth_membership. Each new project also

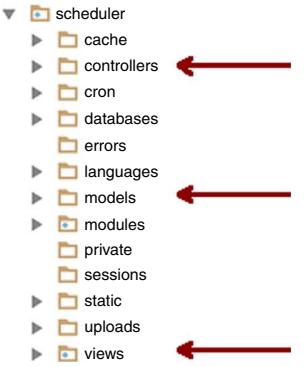

Figure 1.
MVC architecture

comes with a database administration tool that exposes these tables and allows you to list, create, update, and delete records. The database administration tool also creates an import/export feature for each table that you create (Figures 2 and 3).

3.2 The DAL

When you define a table in a models file, you can set attributes such as the data-type, validation requirements, labels, and default values. These definitions generate a database table and they are also used to generate jQuery widgets that are serialized in the views. Below are some screenshots of the model, view, and controller, which I will explain line by line (Figure 4).

Line 54: The table is defined as instructor.

Lines 55-59: Two fields are defined, instructor_name and course. Every table created in web2py has a field that contains an auto-incremented key with the naming convention table_name_id. So for this table an additional field would be created called instructor_id.

Line 60: auth.signature automatically adds four fields to the table, created_by, created_on, updated_on, and updated_by. These fields are then populated each time a transaction occurs with the table, which facilitates the easy joining of a table with the auth_user table.

Line 61: The format command sets how a list of values from the table will be formatted when called for display (Figure 5).

Figure 2.
Interface for table definitions

Figure 3.
Auto-generated import/export functionality

```
54 db.define_table('instructor',
55    Field('instructor_name',
56          label='Enter in the form "last name, first name"'
57          length=255),
58    Field('course',
59          length=255),
60    auth.signature,
61    format = '%(instructor_name)s %(course)s')
62
```

Figure 4.
Instructor table definition in model file

```
61 @auth.requires_membership('manager')
62 def add_instructor():
63     db.instructor.instructor_name.readable=False
64     db.instructor.instructor_name.writable=False
65     db.instructor.instructor_name.default = auth.user.last_name +', ' + auth.user.first_name
66     form = SQLFORM(db.instructor).process()
67     #the_name = db(db.instructor.id==auth.user.id).select(db.instructor.instructor_name)
68     #form.vars.instructor_name = the_name
69     query = db.instructor.created_by==auth.user.id
70     rows = db(query).select(orderby=db.instructor.instructor_name|db.instructor.course)
71     return locals()
```

Figure 5.
Add instructor function in the controller

Line 61: This is a function decoration that requires the user to be a member of the manager security group.

Line 62: This initiates the function definition.

Lines 63-64: When instructor_name is set to readable and writable False, it will not display on the serialized form.

Line 65: The value of the default instructor_name is set to the currently logged on user.

Line 69: A query using the DAL syntax returns all of the courses of the current user.

Line 70: An iterable rows object is created with the values of the select and is returned ordered by instructor_name and course.

Line 71: All of the variables defined in the add_instructor are passed to whatever calls the function (Figure 6).

The add_instructor.html view has the same name as the function in the controller.

Line 1: The extend command includes all of the formatting included in layout.html.

Line 3: The serialized form is displayed at this location.

Line 4: A Python if statement checks to see if the rows object passed from the controller has any values.

Line 13: A Python for statement loops through each value in the rows object.

Lines 16-17: Output shows the instructor_name, and course for the current row.

Line 19: The pass statement takes the place of the indentation which is necessary in Python and closes the for loop.

Line 21: This is a Python else statement.

Line 22: A pass statement closes the outer if statement (Figure 7).

```
{{extend 'layout.html'}}
<h1>Add an Instructor: </h1>
{{=form}}
{{if rows:}}
<hr>
<h4>Your Instructor/Courses</h4>
<Table>
    <tr>
        <td><strong>Instructor Name</strong></td>
        <td><strong>Course</strong></td>

    </tr>
{{for row in rows:}}
    <tr>

    <td>{{=row.instructor_name}}</td>
    <td>{{=row.course}}</td>
</tr>
{{pass}}
</Table>
{{else:}}
{{pass}}
```

Figure 6.
Add instructor view

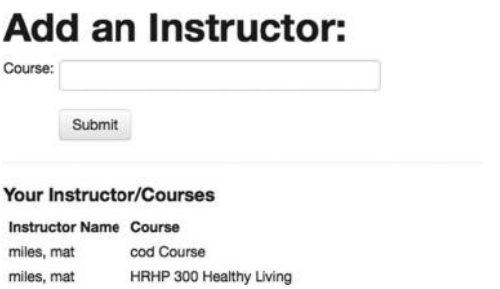

Figure 7.
Add instructor
html view

The screenshot above shows how the html view is presented to the user. A text box is automatically created from the definition in the model. A hidden form value with the author's name is passed from the controller. The rows object populates the list below the form. When the user adds a new course it will post back to itself and the new course will appear in the list below. This is just one example that displays the compact and functionally rich nature of web2py.

4. Menuing

Every web2py application has a built-in menuing structure called menu.py. User functions can be restricted based on permissions. In the example below, there are four menu options: "Upload Assignment," "View Student Assignments," "Edit/Update/Delete Student Assignments," and "Add Instructor/Course." The first option is shown to all users and does not require that the user be a member of any security group. Options two and three require the user to be a member of the Instructors group. The third option requires the user to be a member of the managers group (Figure 8).

In the screenshot below, you can see how this menu is rendered in the html template (Figure 9).

5. Internationalization

The web2py framework was designed from the ground up with internationalization in mind. Many of the standard messages used by any web2py applications are already translated. For new strings you can simply use the "T( )" helper function. For example, if you have the following string "Copy Day Schedule", you can place it inside of the "T( )" helper function as shown below:

< h1 > {{ = T("Copy Day Schedule")}} < /h1 >

Then you can generate a list of strings to be translated in the administration tool (Figure 10).

The next step is to provide a translation for the string in the target language (Figure 11).

In this example we have translated the string into French in the fr.py file. If the user's browser language default is set to French as shown, strings will be displayed to the user in French (Figure 12).

```
response.menu = [
####(T('BYU Idaho Home'), False, 'http://www.byui.edu'),
    (T('Upload Assignment'),False, URL('default','index')),
]
if auth.has_membership('Instructor'):
    response.menu.append((T('View Student Assignments'),False,URL('default','search_instructors')))
if auth.has_membership('Instructor'):
    response.menu.append((T('Edit/Update/Delete Student Assignments'),False,URL('default','manage')))
if auth.has_membership('manager'):
    response.menu.append((T('Add Instructor/Course'),False,URL('default','add_instructor')))
```

Figure 8.
menu.py

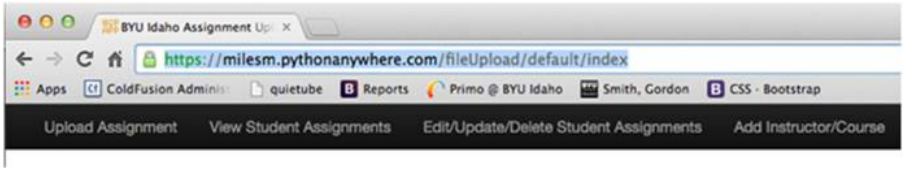

Figure 9.
Menu rendered in html

Now, when we display the html template, we can see the translated string (Figure 13).

As internationalization is built into the architecture of web2py, the process of delivering an application in a variety of languages has been greatly simplified.

6. web2py usage

One of the concerns when a user or organization decides to invest the time in learning a framework is whether there are enough developers using the framework to ensure long-term viability? A review of the literature shows that, especially in the sciences, there is a growing user base, probably due in part to the prevalence of use of the Python language in that discipline. web2py is mentioned as a framework used to develop water resource web applications (Swain *et al.*, 2015). Vanderbilt University used web2py to create a web application to "rapidly visualize a large variety of astronomy datasets of

Figure 10.
Update language strings

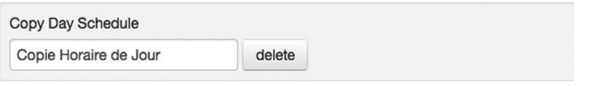

Figure 11.
Translate string

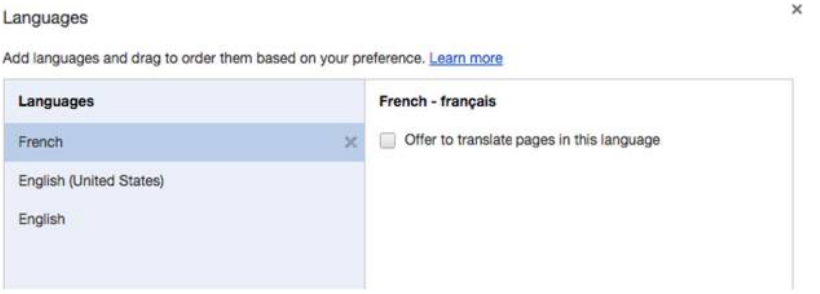

Figure 12.
Set default browser language

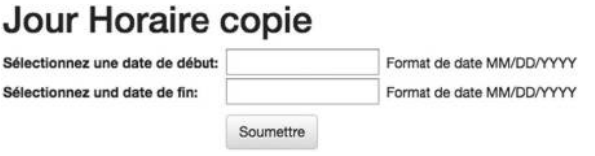

Figure 13.
Copy day schedule translated into French

various formats and sizes" (Burger *et al.*, 2013). It was also the framework of choice to develop an application named KEGG Enriched Network Visualizer that "provides an integrative tool, suitable for users with no programming experience, for the functional interpretation, at both the metabolic and signaling level, of differentially expressed gen subsets deriving from genomic experiments" (Pilalis *et al.*, 2015). Scientists in Singapore and Australia developed an electronic laboratory notebook called CyNote, also built-in the web2py framework (Yong-Yao Ng and Ling, 2010).

7. Conclusion

Though the web2py framework is fairly new, it has shown itself to be a viable option for developing data-driven web applications. It is free and open source, and the community is large enough to support ongoing development. The design of the framework shortens the learning curve and enforces solid programming procedures even for more novice developers. Developers who are already familiar with the Python language will be able to leverage that expertise in Web2py, since the logic is written in Python for both the controllers and templates. No additional languages are required. Academic librarians who have a need to create data-driven web applications to solve local problems may want to consider adding web2py to their technology stack.

References

Burger, D., Stassun, K.G., Pepper, J., Siverd, R.J., Paegert, M., De Lee, N.M. and Robinson, W.H. (2013), "Full length article: filtergraph: an interactive web application for visualization of astronomy datasets", *Astronomy and Computing*, Vol. 2, August, pp. 40-45.

Di Pierro, M. (2010/2013), *web2py: Enterprise Web Framework*, 5th ed., Lulu Enterprises, Raleigh, NC.

Pilalis, E., Koutsandreas, T., Valavanis, I., Athanasiadis, E., Spyrou, G. and Chatziioannou, A. (2015), "KENeV: a web-application for the automated reconstruction and visualization of the enriched metabolic and signaling super-pathways deriving from genomic experiments", *Computational and Structural Biotechnology Journal*, Vol. 13, pp. 248-255.

Swain, N.R., Latu, K., Christensen, S.D., Jones, N.L., Nelson, E.J., Ames, D.P. and Williams, G.P. (2015), "Review: a review of open source software solutions for developing water resources web applications", *Environmental Modeling and Software*, Vol. 67, pp. 108-117.

Yong-Yao Ng and Ling, M.H.T. (2010), "Electronic laboratory notebook on web2py framework", *Python Papers*, Vol. 5 No. 3, pp. 1-13.

Further reading

Di Pierro, M. (2011), "web2py for scientific applications", *Computing in Science and Engineering*, Vol. 13 No. 2, pp. 64-69.

Corresponding author
Mathew Miles can be contacted at: milesm@byui.edu