A Lightweight Web-based Application Framework for Web 2.0 Using Python

ANDY HON WAI CHUN
Department of Computer Science
City University of Hong Kong
Tat Chee Avenue, Kowloon Tong
HONG KONG

Abstract: - This paper provides an overview of the design and architecture of a simple and yet powerful lightweight web-based application framework called the "FOA Application Framework." It follows a Model-View-Controller (MVC) architectural design pattern. The framework is Python-based and provides role-based access as well as a lightweight template engine. The simple syntax of Python makes the FOA framework very easy to use. It operates within a Solaris environment using Apache as the web server. The City University of Hong Kong recently used this framework to create a Web 2.0 AI rostering system for the Equestrian Events of Good Luck Beijing Games that were held in 2007 in Hong Kong in preparation for the 2008 Beijing Olympics.

Key-Words: - Web-based application framework, template engine, Python, Web 2.0

1 Introduction

In today's rapid paced and highly competitive society, when an organization has a business case for a new IT system, it usually does not have time to wait a year or several months for implementation – it would most likely need it within a few weeks or a couple of months. Such was the case in May 2007 when we were commissioned to create a volunteer management system to manage all the volunteers for the Equestrian Events in the Good Luck Beijing Games [3], which were preparation games for the 2008 Beijing Olympics [1, 2]. We had only 6 weeks to design, implement and test the system.

Fortunately, for us, we had a reliable and robust application framework that allows us to quickly create prototypes and refine them in Rapid Application Development (RAD) manner [4]. Our framework is called the "FOA Application Framework." The FOA Application Framework is Python-based [5]. Python is also known to be one of the most productive and easiest to use modern programming languages to date. This further improves the overall efficiency of the project.

This paper describes the overall design and architecture of the "FOA Application Framework." It uses the Good Luck Beijing volunteer management system as an example of how we leveraged features within the framework to quickly and successfully create the system within only a few weeks' time to help Hong Kong pass the International Olympic Committee's readiness requirements for the 2008 Beijing Olympics.

2 Volunteer Management

The volunteer management system for the Hong Kong Equestrian Events is called the "Workforce Management System" (WMS). It is a typical Web 2.0 [6] JavaScript-based rich internet application (RIA) [7, 8] Server-side services were implemented using our own lightweight FOA Application Framework.

The WMS is a secured role-based [9] application with 2 main classes of users – volunteers/staffs, and administrators/planners. Volunteers/staffs have access to WMS features to update their profiles, such as addresses, phone numbers, etc., as well as availability and uniform size information. They can of course use WMS to check their rosters/ schedules, as well as individual/group messages and announcements from the administrators/planners.

Administrators/planners, on the other hand, have access to WMS administrative features to search and retrieve information related to individual volunteers/staffs, define shifts and generate rosters, and send messages/announcements. Figure 1 is an example of a typical "administrator" screen in the

WMS; this particular screen shows part of the roster for one of the divisions. The cells are colored to indicate the first shift the volunteer/staff is assigned to for a particular day. White cells indicate that a volunteer/staff is not available on that day. Red indicates a conflict; either the staff is not available or overlapping assignments were made. Pink indicates that the volunteer/staff is available but not assigned any duties. Mousing over the cell displays details of the assignment. Clicking on any cell will display a form to allow the administrator to edit the assignment.

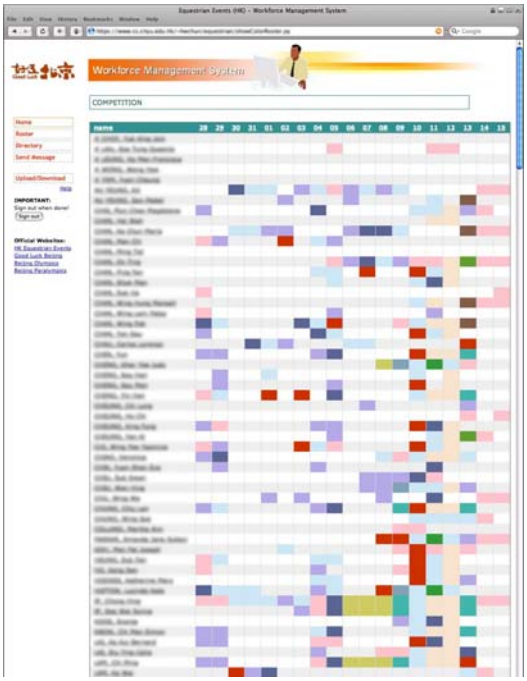

Figure 1. The WMS Roster Screen

3 System Architecture

The system architecture of WMS is similar to other modern Web 2.0 applications. It is based on a distributed multi-tiered Web-based architecture. The client-tier consists of the JavaScript-based RIA client as well as the AJAX-engine. The Web server tier handles the HTTP requests and contains our FOA Application Framework. Dynamic pages are composed using our FOA Template Engine. The application server consists of the domain objects and business logic coupled with an Artificial Intelligence (AI) Engine. The database tier is a persistent store of these domain objects. In addition, the FOA Application Framework maintains a database log of every single system access and action taken. Figure 2 shows the system architecture we used to create the WMS.

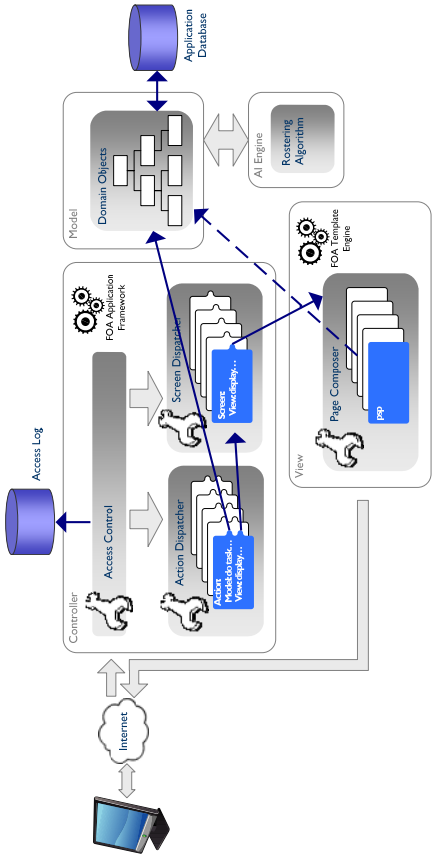

Figure 2. The Architecture of the WMS.

The following describes the design and structure of the FOA Application Framework as well as its underlying FOA Template Engine.

4 FOA Template Engine

The main task for the "FOA Template Engine" is to provide mechanisms to define screen templates for dynamic web pages. Our lightweight template engine operates on top of the Apache [10] web server using the mod_python module [11] that provides Python support and Python Server Pages (PSP) [12]. Python is a modern computer language with a very simple syntax. It is considered by many as one of the more productive and easiest language to use.

However, PSP technology alone only allows one to embed server-side Python code into HTML pages. It does not provide a well-defined mechanism

to define page layout, page components nor dynamic page composition, i.e., how components of a page change depending on desired view or user profile.

Our lightweight template engine allows us to quickly and very easily create a template layout, define the components of the template, and have dynamic content be filled in using PSP. A new screen can be added within minutes, simply by creating a new PSP page (for the dynamic content) and registering the page with our framework. This greatly improves productivity during prototyping and rapid application development.

Our FOA frameworks were created by making heavy use of Python's meta-programming capabilities through the use of the "decorators" [13] language construct, which greatly simplifies application coding. At a very high level, Python decorators may seem similar to meta-programming constructs in other languages, such as Java "annotations" [14] and .Net "attributes" [15]. However, because of Python's dynamic nature, "decorators" are actually quite different from its Java or .Net counterparts.

Syntactically, Python decorators are similar to Java annotations and use the @-symbol. However, Java "annotations" do not directly affect the semantics of a program. .Net "attributes" also perform a "declarative" function and do not modify their referents. In contrast, Python decorators, takes one function and returns another, possibly with modified semantics depending on the dynamic context at the time of use.

In general, Python decorators are usually used to define class/static methods, adding function attributes, tracing, setting pre- and post-conditions, and synchronization. For example, the following illustrates a built-in decorator to define a read-only property of a class:

```
class Bike(object):
  def __init__(self):
    self.__doors = 0
  @property
  def doors(self):
    """Get the number of doors."""
    return self.__doors
```

This will automatically create a "getter" method that allows one to access the "private" __doors attribute for objects belonging to the Bike class. For example:

```
myBike = Bike()       # create a bike object
print myBike.doors    # use getter
```

Python "decorators" allow us to write code in very concise form and have the decorator expand it to more complex programs and with additional supporting functions if needed.

The following illustrates how we made use of decorators to define templates. The "foaTemplatePart" Python decorator creates new components for use by the FOA Template Engine. In the examples below, "footer" is a XHTML component, while "userMenu" references a Python Server Pages (PSP) file. However, the syntax for defining the two different type of web page component is the same.

```
@foaTemplatePart
def footer(req):
  '''<small>Developed by CityU</small>'''
@foaTemplatePart
def userMenu(req):
  '''template/userMenu.psp'''
```

Depending on the page component content, the @foaTemplatePart decorator would either return the desired text string or call the PSP engine to dynamically create content to be used as part of a web page. The structure of the web page itself is defined using our @foaTemplate decorator. For example:

```
@foaTemplate
def myTemplate(req): '''template.psp'''
```

The above would automatically create a function that automatically fills in the template defined in template.psp and return the dynamic page back to the browser. The "template.psp" file itself might look like:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xml:lang="en">
<head>
  <title><% title(req) %></title>
  <% cssjs(req) %>
  <% metatags(req) %>
</head>
<body>
<div id="container">
  <div id="header"><% logo(req) %></div>
  <div id="sideBar"><% menu(req) %></div>
  <div id="mainClm">
```

```
    <% content(req) %>
    <br />
    <br />
    <p class="copyright">
      <% copyright(req) %><br />
      <% footer(req) %>
    </p>
  </div>
</div>
</body>
</html>
```

Our experiments show that we achieve a 90% savings in code size by using our "FOA Template Engine" to define dynamic content.

5 FOA Application Framework

The "FOA Application Framework" provides a framework to create applications following the popular Model-View-Controller (MVC) architecture design pattern [16]. The main function of the "FOA Application Framework" is to act as the "Controller" in the MVC model. The "View" is provided by the previously described FOA Template Engine. And the "Model" is provided by domain objects.

As the "controller," the FOA Application Framework provides three key functions – role-based access control, role-based action dispatching and role-based screen dispatching. The action/screen dispatching mechanism is similar in concept to Jakarta Struts [17].

5.1 Access Control

All access to applications built on top of the FOA Application Framework must go through the "Access Control" module. In other words, each and every click (or page access) will be authenticated by the "Access Control" module. After authentication, roles and privileges will be checked to ensure user has rights to access a particular page or function. All activities are logged into the access control database as audit trail.

Then, depending on the user request, the FOA Application Framework may either route the request to the Action Dispatcher or Screen Dispatcher.

5.2 Role-based Action Dispatching

Within our framework, there are two main types of user requests – a simple dynamic page request, or invoking a REST web service [18]. A dynamic page request does not involve executing business logic or modifying the state of domain objects. This is handled by the role-based screen dispatcher that we will explain later.

The FOA Application Framework supports REST web services, i.e., web service in the form of URI format. The purposes of these web services are usually to create new server-side objects, manipulate existing domain objects and/or possibly modify states of these objects. These REST web servers are called "actions" in FOA.

Actions are also defined using Python decorators. The following illustrates the simple syntax of a typical action in FOA:

```
@actionHandler('admin', next=uploadStatusScreen)
def doUpload(req, **kwargs):
  return manager.doUpload(req, req.form)
```

In this example, the web service "doUpload" can only be accessed by a user with "admin" privileges. The action implementation, i.e. manager.doUpload() is a method provided by the domain "Model." The "next" parameter in the @actionHandler defines the screen flow, i.e. the next screen to display if the action was successfully.

The @actionHandler reduces coding complexity by a 100 times! The three line code shown above is actually replaced by a more complex program that spans 300 lines of Python code. The expanded code includes session handling, timeout control, logging, error handling, and integration with the screen dispatching to display the "next" screen. It also automatically logs any errors and instantly sends an alert, email or SMS, to notify a support personnel if there are any errors during execution.

5.3 Role-based Screen Dispatching

Dynamic screens in FOA are also defined using Python decorators. The following is the code that implements the "uploadStatusScreen" shown in the previous example:

```
@screenHandler('admin', menu=adminMenu)
def uploadStatusScreen(req, **kwargs): pass
```

The code is deceivingly simple. It defines a new dynamic screen that only users with "admin" privilege can access. The "menu" component within the page template is to be replaced with the "adminMenu." This is done behind the scene by the FOA Template Engine. The name of the screen is uploadStatusScreen. However, the function is an empty function, i.e. "pass" only. This is because it

uses the default screen dispatching behavior which is to retrieve screen content from a PSP file with the same name and then inserting it into the layout template that was defined earlier. For richer user experiences, AJAX support is provided through the PSP pages.

The 2 line code shown in the example is automatically expanded to a complex 350 line code using the @screenHandler decorator and thus reduce coding efforts by 99%! Similar to the @actionHandler, the expanded code includes session handling, timeout control, logging, error handling, and integration with template engine as well as automatic support alert if there are any errors during execution.

6 Case Study

The FOA Application Framework and its associated FOA Template Engine were used to create the Workforce Management System (WMS) for the "Good Luck Beijing — HKSAR 10th Anniversary Cup" equestrian games that were held in August 2007. The "Good Luck Beijing" games were held to test the readiness of and prepare Hong Kong for the Equestrian Events to be held in 2008 as part of the Beijing Olympics.

In the "Good Luck Beijing" games, a total of nine delegations comprising some 40 horses participated. To support the games, roughly a thousand volunteers and part-time staff had to be rostered and assigned jobs. This is the role of the WMS.

We only had roughly 6 weeks to design and implement WMS. The first two weeks were used to define the requirements and to perform data preparation, which included creating the HR-XML compliant domain "model." Once the domain model was in place, it only took roughly a week to come up with a detailed second prototype, because of the productivity gains from using the FOA framework and template engine. The scheduling/rostering algorithms were then implemented. The last two weeks were then spent on refining the user interface and interactivity as well as thoroughly testing the application.

Since the system was built on top of our existing Web frameworks and AI libraries, the debug cycle was used mainly to ensure the Web-based process flow matched operational needs to support the equestrian events.

With the help of WMS and our FOA framework, the Hong Kong "Good Luck Beijing" games went very smoothly and Hong Kong was able to successfully pass its Olympics readiness testing [19].

The President of the International Olympic Committee, Mr. Jacques Rogge, commented: *"I am very happy with the preparations. I have spoken with the riders and they are very happy,…Everything is progressing well, and we will have an absolutely fabulous Games here next year."*

The President of the Hong Kong International Olympic Committee, Mr. Timothy Fok, said: *"In spite of the weather, we are satisfied with the results,…We have one year to prepare and I am confident it (the test event) will be good preparation for the Olympics."*

The Chief Judge for the test event, Mr. Martin Plewa, said: *"This is the best test event I've ever been to,… It is almost like [how] the Olympics feel itself."*

The Director of Corporate Administration, Dr. Horace Yuen, commented on CityU efforts on creating the WMS: *"[CityU] was very professional in understanding user requirements… readily available Web-based platform… shortened development time."*

7 Conclusion

This paper presented the architecture and design of our modern Web-based FOA Application Framework and its associate FOA Template Engine. Because of the simplicity of the Python language and its powerful decorator feature, the framework and template engine greatly improves coding productivity and reduces actual code size. Because of the productivity gain, we were able to produce and deploy a sophisticated Workforce Management System, in support of the Hong Kong "Good Luck Beijing" equestrian games within only 6 weeks' time.

References:

[1] The International Olympic Committee Web Site, "Beijing 2008." Retrieved 2008. Available at www.olympic.org/uk/games/beijing/index_uk.asp

[2] The International Olympic Committee Web Site, "Beijing 2008: Rogge In Hong Kong," Retrieved 2008. Available at www.olympic.org/uk/games/beijing/full_story_uk.asp?id=2278

[3] The Hong Kong Equestrian Events Web Site. "Olympic Equestrian – Events." Retrieved 2007. Available at www.equestrian2008.org/eng/olympic_e.aspx

[4] Wikipedia Web Site, "Rapid Application Development." Retrieved 2008. Available at: http://en.wikipedia.org/wiki/Rapid_application_development

[5] Python Programming Language Official Web Site, Retrieved 2008. Available at: http://www.python.org/

[6] O'Reilly, Tim, "What is Web 2.0", Sept 30, 2005. Available at: http://www.oreilly.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[7] Adobe Web Site, "Rich Internet Application," Retrieved 2008. Available at: http://www.adobe.com/resources/business/rich_internet_apps/

[8] Wikipedia Web Site, "Rich Internet Application," Retrieved 2008. Available at: http://en.wikipedia.org/wiki/Rich_Internet_application

[9] Wikipedia Web Site, "Role-based access control," Retrieved 2008. Available at: http://en.wikipedia.org/wiki/RBAC

[10] The Apache Software Foundation Web Site, Retrieved 2008. Available at: http://www.apache.org/

[11] Apache/Python Integration. Retrieved 2008. Available at: http://www.modpython.org/

[12] Python Server Pages. Retrieved 2008. Available at: http://www.modpython.org/live/current/doc-html/pyapi-psp.html

[13] Dr. Dobb's Portal. May 01, 2005, "Python 2.4 Decorators," Available at: http://www.ddj.com/web-development/184406073;jsessionid=TGP3FFMUET5CGQSNDLPCKHSCJUNN2JVN?_requestid=330263

[14] Java Programming Language Documentation, "Annotations," Retrieved 2008. Available at: http://java.sun.com/j2se/1.5.0/docs/guide/language/annotations.html

[15] informIT Web Site, ".NET Reference Guide – Attributes," Aug 19, 2005, Available at: http://www.informit.com/guides/content.aspx?g=dotnet&seqNum=401

[16] Burbeck, S. 1992. "Application Programming in Smalltalk-80: How to use Model-View-Controller (MVC)." University of Illinois in Urbana-Champaign (UIUC) Smalltalk Archive. Available at http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html

[17] IBM Websphere Web Site, "Introduction to Struts Tools," Retrieved 2008. Available at: http://publib.boulder.ibm.com/infocenter/wsphelp/index.jsp?topic=/com.ibm.etools.struts.doc/html/cstruse0001.htm

[18] Wikipedia Web Site, "Representational State Transfer," Retrieved 2008. Available at: http://en.wikipedia.org/wiki/Representational_State_Transfer

[19] The Hong Kong Digest Web Site, "IOC Gives Hong Kong Vote of Confidence," Available at www.hketousa.gov.hk/ny/e-newsletter/07july/Equestrian.htm