**Unisport sample application documentation**

**Problem definition:**

Create a simple application that shows the following data;

/products/

should return the first 10 objects ordered with the cheapest first.

/products/kids/

should return the products where kids=1 ordered with the cheapest first

/products/?page=2

The products should be paginated where page in the url above should return the next 10 objects

/products/id/

should return the individual product.

**Application design:**

Following application consists of four main pages

* Home
* Products
* Kids
* Query data

The layout of the pages are very simple and followed by rapid application development model. The advantage of this model is that get a good result within a very short time span.

**Implementation model:**

Implementation of the model consists of the following steps

* Selection of the tool
* Setting up the project
* Model creation
* Controller creation
* Views creation

This application follows the MVC design pattern which gives us more flexibility and future enhancement with in the application design pattern and coding. In this technique controller plays a key role with in the application and act as a hub all the information flows inward and outward to the controller.

**Selection of the tool:**

While building any project the most difficult part is the selection of the tool. Initially i tried flask and Django both are good tools but web2py web framework attracted me more because of its flexibility and built in ability to create and run the application. Web2py approach is write less code and get more, Web2py is easy to install and run on windows machine. Web2py follow the MVC(model view control convention).

**Setting up the project:**

After downloading and installing web2py version number (2.14.6) I create the application folder unisport which gives me a general layout of the application like default admin user, welcome page etc. Web2py has a built-in feature to import data from \*.csv but in my given task I have given json file so I converted the json file to \*.csv file and from there I imported all data to the table.

**Table structure:**

From json file I extracted the field names and associate the relevant field type and create the table after creating table I imported the all data from result.csv to the table and table was successfully populated. Appendix A shows the structure of the table.

**Main controller creation:**

I created the main controller with the name “uniproducts” and it has several methods their detail is given below, see more details in Appendix B.

**def view():**

view controller select all record and show the products in ascending order with their price since our price field has string data type and contains value like 49,0 so I used replace function to replace , with . and convert them to float and perform the sorting which give me ascending order list of the price field.

**def products():**

This method perform the two functionality one is pagination and other one is sorting the price in ascending order. To do the task I used the request object to get the page response and set the limit as in start and end variable, finally I used the limitby parameter to limit the records which is 10 record per page.

**def kids():**

this method filtered the data where kids=1 and sort them in ascending order by their price and return the record to the view and show the value in table since the price field contain the string data as we so I used the lamda function to replace , with . and convert them to float value and then perform the sorting funtionc.

**def newform():**

when user click on Add link on product page this method helps to create a new form. If you notice in appendix B one line of code help to create a fully functional form and this is the main advantage of using web2py (write less and get more).

**def update():**

update method perform the updation on the specific record which it gets from the request.args(0) object, once we get the id field we can populate the relevant record and this relevant record populated by the form variable so user can see what he is updating, once user click on the submit button form.process().accepted function called and “record updated” message shown on the screen and transfer the view to to the view control which is showing all products.

**def delete():**

delete method also get the value of id field using the request.args(0) object and once it get the id field it also populate the relevant field form, the slightly difference between and update and delete is that under SQLFORM(db.products,record,deletable=True) parameter it has deletable=True parameter which shows the check box at the end of the field and if user checked this box user can delete record and like any delete action, before deleting the record warning dialogue box appeared and if user ignore the warning record successfully deleted and page redirected to the main view page.

**def grid\_products():**

This method is very interesting and one single line of code performed a lot of the functionality(see in appendix B) and shows the real power of the web2py development framework. Sqlform.grid take many arguments and we can customize it as we want and as our need, e.g., paginate=10 only shows the 10 record on the page, orderby clause can ordered any field and so on… it has almost 40 arguments and its very flexible in term of functionality.

Appendix A

**Model of the application**

db.define\_table('products',

Field('is\_customizable', type='boolean'),

Field('delivery',type='string'),

Field('kids',type='boolean'),

Field('name',type='string', requires=IS\_NOT\_EMPTY()),

Field('sizes',type='string', requires=IS\_NOT\_EMPTY()),

Field('kid\_adult', type='boolean'),

Field('free\_porto', type='boolean'),

Field('image',type='string',requires=IS\_URL()),

Field('package', type='boolean'),

Field('price', type='string', requires=IS\_NOT\_EMPTY()),

Field('url',type='string',requires=IS\_URL()),

Field('online\_product', type='boolean'),

Field('price\_old' ,type='string', requires=IS\_NOT\_EMPTY()),

Field('currency',type='string' , requires=IS\_NOT\_EMPTY()),

Field('img\_url',type='string',requires=IS\_URL()),

Field('id',type='integer',requires=IS\_NOT\_EMPTY()),

Field('women\_product', type='boolean'))

Appendix B

**def view():**

rows= db(db.products).select()

pricesort = rows.sort(lambda p: (float(p['price'].replace(',', '.'))))

return locals()

**def products():**

response.title += ' | Pruducts'

if not request.vars.page:

redirect(URL(vars={'page':1}))

else:

page = int(request.vars.page)

start = (page-1)\*10

end = page\*10

rows= db().select(db.products.ALL,limitby=(start,end))

pricesort = rows.sort(lambda p: (float(p['price'].replace(',', '.'))))

return locals()

**def kids():**

records = db(db.products.kids==1).select()

pricesort = records.sort(lambda p: (float(p['price'].replace(',', '.'))))

return dict(records=pricesort)

**def newform():**

form= SQLFORM(db.products).process()

return locals()

**def update():**

record = db.products(request.args(0)) or redirect(URL('view'))

form= SQLFORM(db.products,record)

if form.process().accepted:

response.flash= T('Record Updated')

redirect(URL('view'))

else:

response.flash = T('Please fill the form properly')

return locals()

**def delete():**

record = db.products(request.args(0)) or redirect(URL('view'))

form= SQLFORM(db.products,record,deletable=True)

if form.process().accepted:

response.flash= T('Record deleted')

redirect(URL('view'))

else:

response.flash = T('Please select a check box to delete data at the bottom')

return locals()

**def grid\_products():**

grid = SQLFORM.grid(db.products,paginate=10,buttons\_placement = 'left')

return locals()

**Future enhancement:**

As in future I’m visioning this application as a fully dynamic app, e.g., main screen shows a wizard where user can enter url, or file of any type like Json, xml or csv which further can be converted to the table. Further more, in wizard next step will ask to the user how it wants to represent the data on the screen with few layout examples.

**Testing:**

Web2py provide the built in functionality for testing the controller app and it uses to doctest unit testing technique I tried to apply few test on the controller but I didn’t get the satisfactory results and it took lot of my time because in web2py don’t let you think out of the box in this area I really felt disadvantage of using web2py.

**References:**

<https://stackoverflow.com/>

<http://web2py.com/book>

<http://www.web2pyref.com/example/index>

**Final words:**

After completion of this task I can say that I know a something about python and with confidence I can write “hello word “code. I studied the most of the web2py manual and it gives me very solid and concreate information about python since it was my first time using python so it was little bit scary, but as time passes I feel like home. Now I’m looking forward to do more work in python.