Web Services & API Development Project

HDCLOUD 2015

Emer Thornbury

# Introduction

The aim is to create an application that will allow a user to login either as standard user or an admin user.

Then they will proceed to a display with products, suppliers, cart, item and order.

The display will have a section for each of the above containing a section for the input form to the left and a section for the listing to the right. The listing will refresh after an update of data.

Map of folders and files:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | PythonProject | staticpage | style | scripts | resources |
| Files | App.yaml | Page.html | Main.css | Main.js | Items.xml |
|  | Cartservice.py |  |  | User.js | Products.xml |
|  |  |  |  | Product.js | Suppliers.xml |
|  |  |  |  | Supplier.js |  |
|  |  |  |  | Cart.js |  |
|  |  |  |  | Item.js |  |
|  |  |  |  | Order.js |  |

The app.yaml file in the main folder will specify the libraries and handlers which will call in the cartservice.py file. I will use a jinja html template for display. There will be a main.css file in the style folder. The JavaScript files will be in the scripts folder.

In cartservice.py l will set out the URLs, using an instance of ‘application’. There I will define the ‘webapp2.WSGIApplication’. This will define my API entry points.

The method above provides a very tightly coupled way of implementing the project. That is because the classes and handlers are all in the one python file. If there were a lot of resources in any of the classes this would be a very ‘heavy’ method, as everything is loaded in whether it is used or not.

## Alternate design and constraints:

I would like to use the MVC model as it separates the concerns for best practice and loose coupling. This is the way I would like to do it, the constraint currently is knowledge.

I would also like to JSON and angular JS for managing my data, but as I only currently know a bit of xml I may be forced to use it as there are time constraints.

The above design provides a more loosely coupled API, which would be more horizontally scalable in the event there is a very large amount of traffic. Also it will load faster if all the elements are separated out, rather than in one file. Using Google App Engine also facilitates scalability as a PaaS, which can be used ‘out of the box’ with no configuration and it is built for scalability.

## Security:

The security concerns across the API are that a user must be logged in, there is a check to see if user is logged in, before the GET, PUT and DELETE methods are implemented. The login is managed by GoogleAppEngine, with the option for an admin user. I have chosen not to specify any admin user privileges in this webservice.

## Access:

Access rights are that the user must be logged in.

# The RESTful API

## User:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User | Get | Put (create) | Put (update) | Delete |
| description |  |  | A user can update certain details | A user cannot be deleted. |
| URI | cartservice/get  /users | cartservice/post  /users | cartservice/put  /users | n/a |
| parameters | Id{string}  firstname{string}  lastname{string}  email{string}  address{text} | Id{string}  firstname{string}  lastname{string}  email{string}  address{text} | Id{string}  firstname{string}  lastname{string}  email{string}  address{text} | n/a |
| Pre-condition | Records must exist | User not logged in. No record with that specified id must exist | A record with that specified id must exist. The id and nickname will be populated by system. | n/a |
| Post-condition | N/A | A record with that specified id must exist | A record with that specified id must exist and show changes | n/a |

## Supplier:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Supplier | Get | Put (create) | Put (update) | Delete |
| description | Get the data for a supplier | Create a new supplier, where no data existes for that id. Requires filling all fields | Update an existing supplier and all fields must contain data | Delete an existing supplier, query on id. |
| URI | cartservice/get  /supplier | cartservice/post  / supplier | cartservice/put  / supplier | cartservice/delete  / supplier/(id) |
| parameters | id{string}  nameSupplier{string}  email{string}  phonenumber{integer}  url{string} | id{string}  nameSupplier{string}  email{string}  phonenumber{integer}  url{string} | id{string}  nameSupplier{string}  email{string}  phonenumber{integer}  url{string} | id{string}  nameSupplier{string}  email{string}  phonenumber{integer}  url{string} |
| Pre-condition | Records must exist | No record with that specified id must exist | A record with that specified id must exist | A record with that specified id must exist |
| Post-condition | Records are displayed in list format | A record with that specified id must exist | A record with that specified id must exist and show changes | No record with that specified id must exist |

## Product:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Product | Get | Put (create) | Put (update) | Delete |
| description |  |  |  |  |
| URI | cartservice/get  /product | cartservice/post  / product | cartservice/put  / product | cartservice/delete  / product(id) |
| parameters |  | Fill all required fields | All required fields must contain data | Will be supplied on id query |
| Pre-condition | Records must exist | No record with that specified id must exist | A record with that specified id must exist | A record with that specified id must exist |
| Post-condition | Records are displayed in list format | A record with that specified id must exist | A record with that specified id must exist and show changes | No record with that specified id must exist |
|  |  |  |  |  |

## Cart:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cart | Get | Put (create) | Put (update) | Delete |
| description |  |  |  |  |
| URI | cartservice/get  /cart | cartservice/post  / cart | cartservice/put  / cart | cartservice/delete  / cart(id) |
| parameters |  | Fill all required fields | All required fields must contain data | Will be supplied on id query |
| Pre-condition | Records must exist | No record with that specified id must exist | A record with that specified id must exist | A record with that specified id must exist |
| Post-condition | Records are displayed in list format | A record with that specified id must exist | A record with that specified id must exist and show changes | No record with that specified id must exist |
|  |  |  |  |  |

## Item:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Item | Get | Put (create) | Put (update) | Delete |
| description |  |  |  |  |
| URI | cartservice/get  /item | cartservice/post  / item | cartservice/put  / item | cartservice/delete  / item(id) |
| parameters |  | Fill all required fields | All required fields must contain data | Will be supplied on id query |
| Pre-condition | Records must exist | No record with that specified id must exist | A record with that specified id must exist | A record with that specified id must exist |
| Post-condition | Records are displayed in list format | A record with that specified id must exist | A record with that specified id must exist and show changes | No record with that specified id must exist |
|  |  |  |  |  |

## Order:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Order | Get | Put (create) | Put (update) | Delete |
| description |  |  |  |  |
| URI | cartservice/get  /order | cartservice/post  / order | cartservice/put  / order | cartservice/delete  / order |
| parameters |  | Fill all required fields | All required fields must contain data | Will be supplied on id query |
| Pre-condition | Records must exist | No record with that specified id must exist | A record with that specified id must exist | A record with that specified id must exist |
| Post-condition | Records are displayed in list format | A record with that specified id must exist | A record with that specified id must exist and show changes | No record with that specified id must exist |
|  |  |  |  |  |