SE31520: ‘MyAlcoholFreeWine.com’

Due on Monday, December 7, 2015

James Euesden - jee22

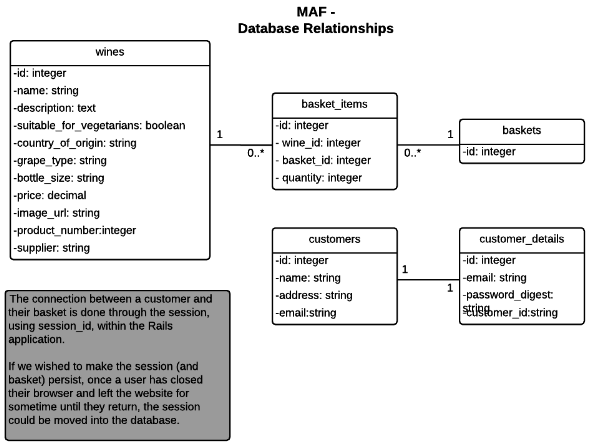
Contents

Introduction

I had to do this [1]

MAF Architecture

Design



The connection between a customer and their basket is done through the session, using session_id, within the Rails application.

If we wished to make the session (and basket) persist, once a user has closed their browser and left the website for sometime until they return, the session could be moved into the database.

Figure 1: The intended design of the database and relations between tables in the MAF application.

Justify why you built it this way What is a session? Basket Id and Customer Id for use in login and basket while they are there How do you get the Wines? Wines stored with supplier - Supplier in config for if it was deployed, but could also be in Database. Which is better? Thin Controllers and models - Keep logic seperate, DRY principles?

MVC

```
can have code here
```

Web Service Architecure

Simple web service has two RESTful routes, Orders and Wines. Get Wines with /wines.json, and send Orders with /orders. As RESTful resources, we don’t need to direct to a route like ’send_order’ or ’get_wines’. Just making a GET request on the wines.json returns us the Wines (since last updated, based on headers), and sending a POST to /orders with the json data for a new order automatically routes to ’create’ in the Orders controller. Just adds the order as a record into the database.

Test Strategy

System Testing

MAF System Test Table

ID	Requirement	Description	Inputs	Expected Outputs	Pass/Fail	Comments
A1.1	FR1	Customer can browse all cheapest wines in paginated (alphabetically ordered) list	n/a	All cheapest wines from the suppliers are displayed in alphabetical order, and are paginated.	p	
A1.2	FR1	Customer can browse all cheapest wines in paginated (alphabetically ordered) list: When a wine is updated, we see this reflected in the wines list	n/a	The wines list is updated to relfect the change of wine(s) in the supplier, e.g. the description of a wine has been updated	p	
A1.3	FR1	Customer can browse all cheapest wines in paginated (alphabetically ordered) list: When a wine a cheaper wine by a different supplier is found, display this wine in the wine list as the cheapest, no longer showing the previous wine by the other supplier.	n/a	The wines list is updated to show the new cheapest wine by the alternate supplier	p	
A2.1	FR2	A customer can Search for any Wine by full word, and any Wine that has any information that matches that search will be returned.	The search term request	The Wines that match the search term request, be it from name, description, country of origin, etc, will be shown	p	
A2.2	FR2	A customer can Search for any Wine by partial word, and any Wine that has any information that matches that search will be returned.	The search term request	Any Wines that match the partial term will be returned and shown to the customer.	p	
A2.2	FR2	A customer Searches for text that is not associated with any wine	The search term request	No Wines will be returned, and the user is informed that their search returned no results	p	
A3.1	FR3	Display detail of a Wine when selected by a User	The wine to be viewed	The full information about a wine will be shown Name, Description, Price, Supplier, Country of Origin, Bottle Size, an image of the bottle, Grape Type and if it is Suitable for Vegetarians.	p	
A4.1	FR4	A Customer can add a Wine to their Basket from the Wine view	The wine to be added to the Basket	The Wine they selected to be added to the Basket is added to their Basket	p	
A4.2	FR4	A Customer can add a Wine to their Basket from the full Wines list view	The wine to be added to the Basket	The Wine they selected to be added to the Basket is added to their Basket	p	
A4.3	FR4	A Customer can specify the quantity of the Wine they wish to purchase and then add that amount of the Wine to their Basket	The wine to be added to the Basket and the quantity of the Wine	The Wine they selected to be added to the Basket is added to their Basket, in the quantity they selected to add to basket	f	Quantity of Wine to add to Basket not implemented for this prototype, only one of the Wine selected may be added to the basket on the button click, but a Customer may have as many of that Wine as they wish in the Basket.
A5.1	FR5	Display Shopping Basket Shopping basket empty	n/a	Do not display an empty basket	p	
A5.2	FR5	Display Shopping Basket Wines displayed in shopping basket that the user has added to their basket, with the quantity.	Basket_items that the customer has placed into their Basket.	Display the Basket with the customers selected wines and their quantities	p	
A5.3	FR5	Display Shopping Basket Remove a wine from the shopping basket.	Items in the basket to be removed.	Display the Basket with the customers selected wines and their quantities, and remove an item when 'Remove' is clicked beside a Basket Item. A notification will inform the user that their Basket Item has been removed.	p	
A5.4	FR5	Display Shopping Basket Empty the entire shopping basket.	Items in the basket to be removed.	All items are removed from the Basket and no Basket will be displayed. A notification will inform the user that their Basket has been emptied.	p	

A5.6	FR5	Display Shopping Basket Change the quantity of the wines in the shopping basket	Items in the basket to have quantity changed, and quantity differences.	A Basket Item can have its quantity changed	f	Out of scope for prototype requirements.
A6.1.1	FR6a	A logged-in Customer may proceed to Checkout and place an order for items in their Basket	The customer is logged in and has Wines in their Basket they wish to purchase	The order is sent to the Wine Suppliers of the Wines the user wishes to purchase	p	
A6.2.1	FR6b	A Customer who is not logged in is prompted to Login before they can Checkout and purchase the Wines in their Basket	The Wines in the Basket the Customer wishes to purchase	The customer is prompted to login when they click to Checkout, and once logged in are returned to their Basket view to try Checkout again.	p	
A7.1	FR7	A Customer who is not logged in can use their email and password to login to their account	The customers email and password	The Customer is logged in and given a notification that they are now logged in. The user who is currently logged in is displayed in the header of the website	p	
A7.2	FR7	A Customer who is not logged in attempts to login with bad credentials (unknown email/password)	The customers email and password (invalid)	The Customer is informed that their email/password was invalid and is not logged in	p	
A7.3	FR7	A logged in Customer may click logout to exit from their account (and also remove their session/basket)	The session must be aware of the current customers basket and id	The Customer is logged out of their account and informed that they have been logged out	p	Note: In the current implementation, the Basket is not persistent once a user logs out. In a future version, once the Customer is logged out it is still possible to retain their Basket for their next login visit. Out of scope for prototype.
A7.4	FR7	A Customer who is not logged in is able to make a new account The account being created is using an email address not yet known to the system	The Customer details, email and password	The Customer account is created, they are told it has been made and are requested to login	p	
A7.5	FR7	A Customer who is not logged in is able to make a new account The account being created is using an email address already known to the system	The Customer details, email and password	The Customer is informed that the email address is already in use and they must supply a different email address	p	
A7.6	FR7	A Customer who is not logged in attempts to make a new account The Customer fails to supply some of their details or email	The Customer details, email and password, with some information left blank	The Customer is informed that they have not supplied some required information, and asked to try again.	p	
A7.7	FR7	A Customer who is not logged in attempts to make a new account The supplied confirmation password does not match the given password	The Customer details, email and password, with wrong confirmation password	The Customer is informed that their password and confirmation password do not match and they must try again	p	

Unit Test for models and controllers

Demonstrate implementation of own Unit Tests for models and controllers (show test executions passing, show test names) Screenshots of tests passing

Self-Evaluation

Mark Breakdown

Screencast

Showed 2 Web Service Suppliers running, showed MAF running with all functionality discussed in this report. Expanded on this to show tests passing, Search in depth, etc Showed orders being received by the web suppliers Mark: /10%

Design

Design diagrams for both how I expected MVC to look and also for my database ideas, along with improvements. Explanation for my designs present and meeting the requirements, backed up through screencast. Documentation quality is okay, could be more in-depth, and design could be improved, but I’m new to Rails and need more experience. Mark: /20%

Implementation: MAF

Application runs, and meets the requirements (except quantity indicator when adding to basket). Rails controllers make calls to models, and views receive the data from the controllers in order to display it. Code is commented, attributions to code where used and identifier names speak for themselves. Mark: /25%

Implementation: Web Service

Web Services run, code has comments and names are good. RESTful resources, wines and orders, can call them and the routing sends the request to the controller through the type of request. Mark: /15%

Testing

Unit tests for each controller, more than just the ones generated by rails, and tests for the models, validations and own methods too. System test table matches assumed customer expectations. Results are discussed and based on these what could be taken forward if this prototype were to be fully implemented Mark: /15%

Evaluation

Full evaluation of report, MAF and web service, with summary of evaluation below the mark breakdown, with what was learned, difficult and easy. Mark: /5%

Flair

Implementation of Search using Solr with Sunspot to search partial words from all sections, editing the schema to change the gram of the search. Calling the WebService asynchronously with SuckerPunch, scheduling it to update with rufus-scheduler for hands off updating without interrupting the customer as they browse the website. Mark: /10%

Total

Mark: /100%

Summary

I think I met all of the functional requirements as requested (bar quantity indicator in FRwhatever), and the resources as RESTful and data is represented in the Rails way. I know there are areas for improvement (security, clean up, persistent sessions and baskets, better tests, cucumber tests, etc), but as far as the time allowed, my understanding and learning of Rails and this assignment tasking me to just create a prototype, I am happy with the resulting program, and happy I could learn more about Rails and Ruby, how to implement an MVC pattern, talking between web services and even implement external features, like learning configurations of Solr for my search. I found the most difficult part to be implementing routing, and getting my links in the views to exectue the correct controller actions as I wanted (for the more complex tasks, such as removing the last item from a basket, logging in and sending the webservice order). Before this assignment I had little working knowledge of rails outside of the workshops and a small amount of experience with routing, and doing this assignment helped me get my head around routing and how it works, and has helped me feel comfortable going forward in the future to building more Rails apps when I want to make a web application.

References

[1] Chris Loftus, "MyAlcoholFreeWine.com",SE31520/CHM5820 Assignment 2015-16, October 27 2015