

CS 5551 Second Increment Report

Adam Carter

Class ID: 9, Project Group ID: 1

CS 5591 – Spring 2015

I. Introduction

This is the summary report of the second iteration of work performed on the Terrapin Collection Manager system. This system proposes to implement a framework for organizing and managing a user's Board Game collection. The full background of this project can be found in the Project Proposal Document [1] submitted previously.

For this second iteration, my primary focus to begin the iteration was on the agent process and improving the process for fetching and then approving game content. While not complete, this process took longer than expected due to a high degree of variation and inconsistency in the data. Focus then shifted to the Services tier, where User and Collection services were implemented. Finally, the last few days involved finally performing some front end work and creating an initial set of web pages.

Since many of the details behind the architecture of this system were discussed previously in my Project Plan [2] and elaborated on during the First Iteration [3], I will not be repeating elements of those discussions here except to discuss changes or additions.

II. Objectives

The objectives of this iteration, given where last iteration ended, were to solidify the external data pull operations, prepare that data to be analyzed and converted into Game objects, then associated the other external sources via matching to the Game.

Roughly the first half of this iteration was devoted to this process, with most of the code changes coming in the Agent layer, supported by corresponding changes to the ac-games-pojo project as new issues were discovered. To put some numbers into perspective:

Total number of Games listed on BoardGameGeek I am tracking: 158536

Total number of Vendor entries I am tracking: 146429

Because of the scope, I spent a fair amount of time working on building some intelligence into the processing effort, determining which entries were invalid or incomplete, which items are capable of being ignored (for example, entries for game supplies instead of games), and attempting to automatically process and confirm as many entries as possible. This effort is still in flight, but so far I have been able to review more than 93% of BoardGameGeek entries and developed a means of categorizing Vendor data that should help during the automated matching work.

Since some of that work can only be done manually, I turned my attention to the UI, rather than try to do all that work by hand. However, prior to being able to work on that interface, I realized I needed to build out the services to support not just Game and GameReltn data, but

also Users to support login and permissions, which then led to the need to implement Collection data to complete the User object and its child classes. Most of the second half of this iteration was spent building out the required REST service endpoints, which will be discussed later.

Finally, the last four days I have been finally diving into HTML5 and Bootstrap to work on developing a UI. I was able to create a home page, about page, and login and signup pages to move forward, but those pages allowed me to learn a lot more about HTML5 and Bootstrap in particular. This time will be very valuable moving forward in the remaining iterations.

III. Design Overview

Fundamentally, there have been no significant changes to the overall architecture of the system. This iteration was really more about expanding capabilities than exploring design issues, with the exception perhaps of the web pages. Much of the planning on how new services should be implemented has been done previously.

One change I did make was to change the structure of my web pages to move away from a single page model where multiple pages are specified as page content in the HTML body and instead created separate HTML pages for each required page. There were issues I encountered related to how the page separation was being handled (or not handled). This forced me to consolidate libraries, generating a single common .css file for all related pages, and has resulted in some level of duplicated code, but has the tradeoff of being cleaner to manage events and reduced page load times due to javascript tasks that are required in some instances to run at page load.

Stories

Story ID	Description	Type	Priority	Comments	Tasks	Owner
#32	As the system, I need to be able to read and write Game information to the database so I can build an index of Game content I need to be able to read and write Game data objects to the database	Task	Green	0	Done	apshaiTerp
#33	As the system, I need to be able to read and write Game Relation information to the database so I can build an index of Game relationships	Task	Blue	0	Done	apshaiTerp
#14	SPIKE: Research user creation and authentication requirements This story may not be required, but I'm adding a placeholder for this story here.	Task	Yellow	0	Done	apshaiTerp
#15	As a User, I need to be able to login to the system so I can access personalized or protected content	Task	Orange	0	Done	apshaiTerp
#16	As the system, I need to be able to access user information so I can access items specific to my personal game collection This story applies to the /user service root. This service is required before ancillary services that require user updates can be implemented.	Task	Purple	0	Done	apshaiTerp
#13	As a User, I need to be able to create an account so I can access personalized content This should cover both the roles of Administrative and Collection Users. There may need to be a SPIKE story added before this to research authentication techniques.	Task	Red	0	Done	apshaiTerp
#30	As the system, I want to proactively update external data I have cached so that I can stay current with that data, especially if users have notifications requested for a given game.	Task	Orange	0	Reviewing	apshaiTerp

From a story perspective, 6 stories were completed, with one close to completion, and another 5 stories were begun, but not completed, which will be rolled into the third iteration.

#30 As the system, I want to proactively update external data I have cached so that I can stay current with that data, especially if users have notifications requested for a given game.

Reviewing Tasks | 0 Comments Services #E1 apshaiTerp

#11 As an Administrative User, I need to be able to define relationships between games and price data so that users can have access to retail information without having to log in directly to the retailer website.

This story should cover the ability to associate any Price Data category to a game so users can quickly access price information for this game.

Doing Tasks | 0 Comments Admin CSI MM AdminClient #E2 apshaiTerp 5

#6 As an Administrative User, I need to be able to view new BoardGameGeek content so I can verify accuracy prior to conversion into Game content.

Some elements of the BoardGameGeek game data may need to be reviewed for accuracy before the entry is converted in to a Game object in our system. This story should cover the ability to view this content.

Doing Tasks | 1 Comment BGG Admin AdminClient #E2 apshaiTerp 5

#7 As an Administrative User, I need to be able to view new CoolStuffInc content so I can verify accuracy prior to conversion into Game content.

Doing Tasks | 0 Comments Admin CSI AdminClient #E2 apshaiTerp 2

#8 As an Administrative User, I need to be able to view new MiniatureMarket content so I can verify accuracy prior to conversion into Game content.

Doing Tasks | 0 Comments Admin MM AdminClient #E2 apshaiTerp 2

#10 As an Administrative User, I need to be able to submit verified Game content so I can build the index that will drive the collection management tool.

Doing Tasks | 0 Comments Admin BGG AdminClient #E2 apshaiTerp 5

The backlog currently contains another 15 items. Many of those stories are related to the front-end HTML pages that will need to be developed that could not even be attempted without the services implementation I have been working on thus far, and I expect many of those tasks to be accomplished reasonably quickly, so at this point, I still feel confident I will be able to deliver on all the stories planned.

Finally, on the topic of design for the HTML pages, I have settled on some common elements, that I intend to use to bind the Look and Feel of my site together. There will be a navbar element at the top of the page, followed by a large visual component referred to by Bootstrap as a Jumbotron. There are also common themes of Terrapin images and a background image that all pages share that is defined in the common .css file to be shared by my HTML pages. Examples of this will be shown later in the Implementation discussion.

IV. Implementation Details

To begin presenting what has been implemented during this iteration, I want to present some quick statistics that were generated to give an idea of progress made. Eclipse has a number of plugins that can generate code statistics driving off any number of criteria. For clarity, I will be presenting statistics in the following 4 categories: Number of Classes, Number of Tests, Total Lines of Code (TLOC), and Meaningful Lines of Code (MLOC – A measurement that ignores whitespace and commented lines).

Project	Classes	Tests	TLOC	MLOC
ac-games-pojo - iteration 1	22	28	1777	1086
ac-games-pojo - iteration 2	38	42	2713	1568
Increase during second iteration	16	14	936	482
ac-games-db - iteration 1	3	0	54	39
ac-games-db - iteration 2	3	0	103	117
Increase during second iteration	0	0	49	78
ac-games-db-mongo - iteration 1	7	3	2841	2419
ac-games-db-mongo - iteration 2	13	9	3200	2753
Increase during second iteration	6	6	359	334
ac-games-restservice-spring - iteration 1	7	4	900	725
ac-games-restservice-spring - iteration 2	22	9	2927	2321
Increase during second iteration	15	5	2027	1596
ac-games-agent - iteration 1	6	0	640	463
ac-games-agent - iteration 2	13	0	1718	1409
Increase during second iteration	7	0	1078	946
Overall System Numbers - iteration 1	45	35	6212	4732
Overall System Numbers - iteration 2	89	60	10661	8168
Increase during second iteration	44	25	4449	3436

This data summary illustrates clearer than a lengthy discussion how progress has been made in all major project components. There was more work done in the Mongo space than is truly reflected in the line counts, since I was able to simplify a handful of operations by refactoring and genericizing a handful of calls.

The REST service grew the most from a raw amount of code perspective. During this iteration I implemented the following new REST service endpoints with full CRUD support (GET, PUT, POST, DELETE) unless otherwise indicated.

```
/game
/gamereltn
/user
/collection
/collectionitem
```

/login (Only supports POST for performing login operation, and PUT for updating sensitive user data)

All 6 of these new services, with their corresponding data objects and validation requirements were done this iteration.

The only major implementation detail that differs from earlier implemented services is that in some cases there are cascading effects of some data changes, for example, if I remove a User from the system, I must also remove his collection and every item in his collection from the system as well.

For reference, the code repositories for each project are listed below. All code for this release has been checked in to the Master branch of each project. Because class diagrams at this point are a little large to make inclusion in this document meaningful, they have been uploaded as part of the github project under the src/model folders.

- **ac-games-pojo**s – <https://github.com/apshaiTerp/ac-games-pojo>
- **ac-games-db** – <https://github.com/apshaiTerp/ac-games-db>
- **ac-games-db-mongo** – <https://github.com/apshaiTerp/ac-games-db-mongo>
- **ac-games-restservice-spring** – <https://github.com/apshaiTerp/ac-games-restservice-spring>
- **ac-games-agent** – <https://github.com/apshaiTerp/ac-games-agent>

Finally, the last element I want to discuss is the HTML development. All development for this project is being checked in to <https://github.com/apshaiTerp/agent-site>. Included below are some screen shots from both the Standard Web view, as well as Mobile.

The screenshot shows a web browser window for the Terrapin Collection Management system. The main header features a large image of a turtle with a mustache, with the text "You can trust a Terrapin with a mustache!". Below this are three smaller images: one of a room full of board games, one of many turtles, and one of shelves filled with board games. The page is divided into several sections with headings and descriptions:

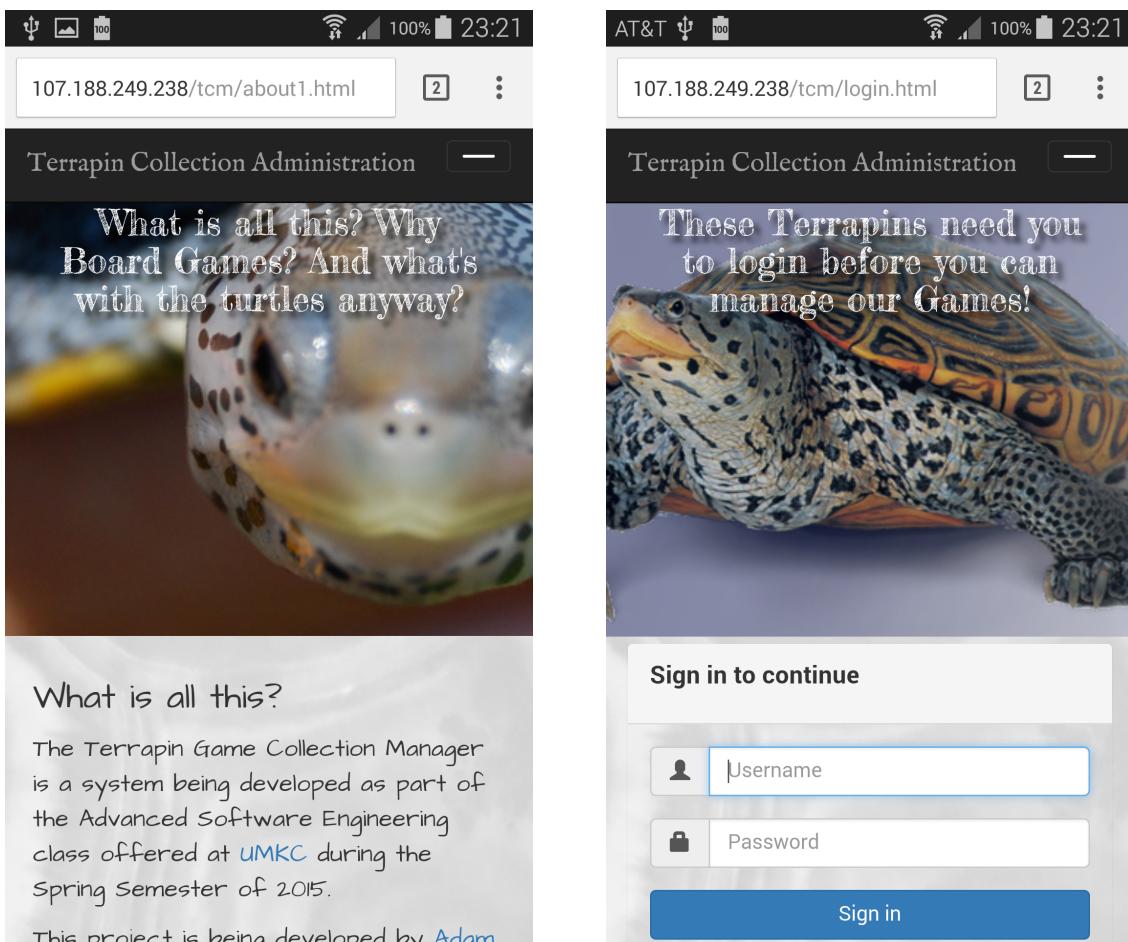
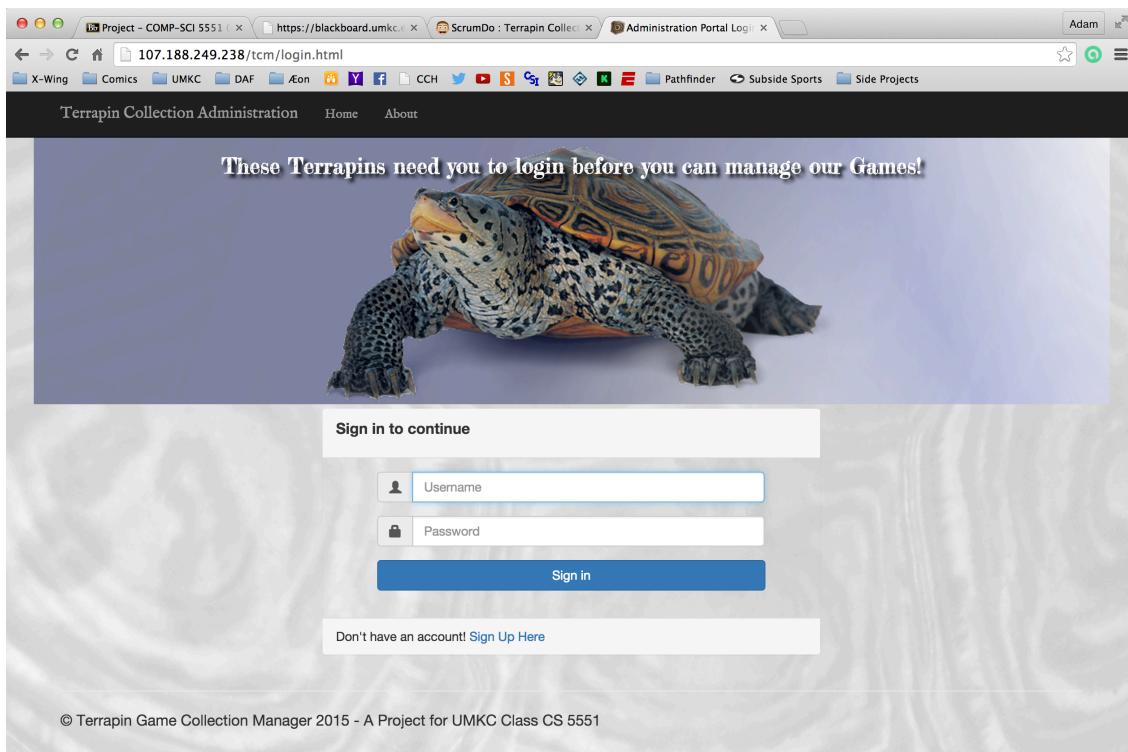
- Too Many Games?**

The Terrapin Game Collection Manager can help bring order to the chaos. You may not have enough shelf space to fit it all, but with this tool you'll always know what you have. Generate recommendations for your game night, keep tabs on new games in your wish list, and tweak game details like player count or playing time to better help you use your collection.
- Too Many Voices?**

The Terrapin Game Collection Manager can help you remember those tutorial videos, game manuals, or FAQs you want to find without having to wade through the noise of other sites. Store just those things you want to find, and your game nights will go that much smoother.
- Where To Buy?**

The Terrapin Game Collection Manager helps track some of the most common online Board Game vendors and makes it easy for you to compare prices and track availability. It also allows you to add links to other vendors you might be watching so you can see it all in one place.

At the bottom, there is a large image of a surprised turtle, a "Surprised?" heading, and a "Sign Up Now" button.



As the screen shots show, by using Bootstrap I was able to implement a responsive design that if not always as clean as the full browser version, is functional and scales all the elements appropriately.

V. Testing

Unit testing was expanded quite a bit with this iteration, though the ac-games-agent process is still without Unit Test.

The important tests I focused on were ensuring that objects behaved correctly, that the REST calls were processed correctly, and that the database operations processed data correctly. All of those tests have been implemented. As with the last iteration, I am including the surefire and clover reports, which have also been uploaded to my documentation site for this iteration. The Code Coverage percentages are lower than the true test coverage numbers because the tests aren't all running through a single source. It is possible to do it this way, but the maven configuration necessary has not been a priority.

- ac-games-pojo

Surefire Report

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
42	0	0	0	100%	0.542

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[Summary] [Package List] [Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
com.ac.games.data.mock	42	0	0	0	100%	0.542

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

com.ac.games.data.mock

	Class	Tests	Errors	Failures	Skipped	Success Rate	Time
	TestBGGParser	14	0	0	0	100%	0.45
	TestCoolStuffIncParser	14	0	0	0	100%	0.035
	TestMiniatureMarketParser	14	0	0	0	100%	0.057

POJO library for Collection Management project 0.2.0-SNAPSHOT Clover Coverage Report

Clover Coverage Report - POJO library for Collection Management project 0.2.0-SNAPSHOT
Coverage timestamp: Wed Mar 18 2015 14:01:37 CDT

App Test Results Clouds
Overview Package File
FRAMES NO FRAMES SHOW HELP

	Packages	% Filtered	Average Method Complexity	Uncovered Elements	TOTAL Coverage	
Clover database Wed Mar 18 2015 14:01:35 CDT	4	0%	2.15	1,084	43.6%	

Package	Files	% Filtered	Average Method Complexity	Uncovered Elements	TOTAL Coverage	
com.ac.games.rest.message	2	0%	1	28	0%	
com.ac.games.data	28	0%	1.85	1,011	29%	
com.ac.games.data.parser	3	0%	25.5	45	90.4%	
com.ac.games.exception	1	0%	1	0	100%	

Report generated by Clover Code Coverage v3.3.0 | Wed Mar 18 2015 14:02:01 CDT | Clover: Commercial License registered to Cerner Corporation.

POJO library for Collection Management project 0.2.0-SNAPSHOT Clover Coverage Report

Clover Test Report - POJO library for Collection Management project 0.2.0-SNAPSHOT
Coverage timestamp: Wed Mar 18 2015 14:01:37 CDT

App Test Results Clouds
Overview Package File Test
FRAMES NO FRAMES SHOW HELP

Project	Tests	Fail	Error	Time	% Pass
Clover database Wed Mar 18 2015 14:01:35 CDT	42	0	0	0.564	100%

Packages	Tests	Fail	Error	Time(secs)	% Pass
com.ac.games.data.mock	42	0	0	0.564	100%

Report generated by Clover Code Coverage v3.3.0 | Wed Mar 18 2015 14:02:01 CDT | Clover: Commercial License registered to Cerner Corporation.

Surefire and Clover reports for this project are stored under <https://github.com/apshaiTerp/cs-5551-documentation/tree/master/maven%20sites/ac-games-pojos/ac-games-pojos-0.2.0-SNAPSHOT%20site>.

- **ac-games-db-mongo**

For this package, the same basic testing strategy was used as was done in the first iteration: Create a series of 3 objects, run a set of inserts and updates, verify that the correct results were obtained for all operations as anticipated. This was done for all new database objects.

Surefire Report

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
9	0	0	0	100%	3.012

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[Summary] [Package List] [Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
com.ac.games.db.test	9	0	0	0	100%	3.012

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

com.ac.games.db.test

	Class	Tests	Errors	Failures	Skipped	Success Rate	Time
	TestMongoGamesDatabase	9	0	0	0	100%	3.012

MongoDB Database Implementation 0.2.0-SNAPSHOT Clover Coverage Report

[Dashboard](#) [Coverage Reports](#) [Coverage \(Aggregate\)](#) [Test Code \(Aggregate\)](#) [Test Results](#)

Application Packages
com.ac.games.db (75%)
com.ac.games.db.mongo (70.7%)

Classes	Tests	Results
Class		Coverage
BGGGameConverter		(80%)
CSIDataConverter		(82.5%)
CollectionConverter		(75.3%)
CollectionItemConverter		(81.1%)
GameConverter		(77.7%)
GameReltnConverter		(76.6%)
MMDataConverter		(79.5%)
MongoDBFactory		(75%)
MongoGamesDatabase		(64.1%)
UserConverter		(67.4%)
UserDetailConverter		(73.4%)

Clover Coverage Report - MongoDB Database Implementation 0.2.0-SNAPSHOT
Coverage timestamp: Wed Mar 18 2015 14:04:58 CDT
[Overview](#) [Package](#) [File](#)
[FRAMES](#) [NO FRAMES](#) [SHOW HELP](#)

Statistics for project Clover database Wed Mar 18 2015 14:04:53 CDT:
Stmts: 1,368 LOC: 3,118 Total cmp: 663 Stmt/Meth: 10.21
Branches: 812 NCLOC: 1,719 Cmp density: 0.48 Methods/Class: 12.18
Methods: 134 Files: 11 Avg method cmp: 4.95 Classes/Pkg: 5.5
Classes: 11 Packages: 2

Coverage 11 classes, 1,636 / 2,314 elements
70.7%

Test Results 9 / 9 tests 2.55 secs
100%

Class Coverage Distribution
Classes
0% Coverage 100%

Top 11 Project Risks
MongoGamesDatabase UserConverter UserDetailConverter
GameConverter BGGGameConverter
MMDataConverter CSIDataConverter MongoDBFactory
CollectionConverter GameReltnConverter CollectionItemConverter

Coverage Tree Map
com.ac.games.db.mongo

MongoDB Database Implementation 0.2.0-SNAPSHOT Clover Coverage Report

Clover Test Report - MongoDB Database Implementation 0.2.0-SNAPSHOT
Coverage timestamp: Wed Mar 18 2015 14:04:58 CDT

Overview Package **File** Test
FRAMES NO FRAMES SHOW HELP

Class	Tests	Fail	Error	Time (secs)	% Tests Success
TestMongoGamesDatabase	9	0	0	2.551	100%

Tests	Started	Status	Time (secs)	Message
TestMongoGamesDatabase.testCollectionSimple	18 Mar 14:04:57	PASS	0.273	
TestMongoGamesDatabase.testCSIData	18 Mar 14:04:57	PASS	0.273	
TestMongoGamesDatabase.testMMData	18 Mar 14:04:58	PASS	0.275	
TestMongoGamesDatabase.testGameData	18 Mar 14:04:56	PASS	0.274	
TestMongoGamesDatabase.testCollectionItems	18 Mar 14:04:57	PASS	0.277	
TestMongoGamesDatabase.testBGGData	18 Mar 14:04:55	PASS	0.357	
TestMongoGamesDatabase.testGameReinData	18 Mar 14:04:56	PASS	0.27	
TestMongoGamesDatabase.testUserDetails	18 Mar 14:04:57	PASS	0.277	
TestMongoGamesDatabase.testUsers	18 Mar 14:04:56	PASS	0.275	

Surefire and Clover Reports for this project are stored under <https://github.com/apshaiTerp/cs-5551-documentation/tree/master/maven%20sites/ac-games-db-mongo/ac-games-db-mongo-0.2.0-SNAPSHOT%20site>.

- **ac-games-restservice-spring**

As with the first iteration, testing was implemented using the Spring Framework and the Rest Assured Mock API. Like with the first iteration, I attempted to take a single object through all the calls for that REST endpoint. This includes in more complicated cases like new user creation checking all objects created by that process.

I discovered during this process that there are some limitations to the Mock framework, especially when it comes to data validation. For example, the Rest Assured framework does not correctly process Date values. Instead of actually converting the value to a Date, it displays the millisecond value represented by that Date. This is a limitation only of the test harness, as I can verify through non-mock usage that actual Date values are being passed through the call. Another limitation is the lack of Enum handling. Several of my report objects use Enum values. The framework is automatically casting Enum values into the `toString()` literal value, which made initially testing some objects difficult.

Finally, a shortcoming in current test is that most of my tests demonstrate the Happy path use case, even though there is robust parameter validation and error handing in the REST controllers. It was not deemed a prudent investment of time to implement all the edge case executions as tests at this time, since I have the virtue of being the only user and developer of this system. Had this not been the case, these would have been given a higher priority. That said, the tests as implemented still are sufficient to ensure all REST-accessible methods execute as expected.

Surefire Report

Summary

[Summary] [Package List] [Test Cases]

Tests	Errors	Failures	Skipped	Success Rate	Time
9	0	0	0	100%	10.796

Note: failures are anticipated and checked for with assertions while errors are unanticipated.

Package List

[Summary] [Package List] [Test Cases]

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
com.ac.games.rest.test	9	0	0	0	100%	10.796

Note: package statistics are not computed recursively, they only sum up all of its testsuites numbers.

com.ac.games.rest.test

Class	Tests	Errors	Failures	Skipped	Success Rate	Time
BGGControllerTest	2	0	0	0	100%	3.834
CollectionControllerTest	1	0	0	0	100%	0.731
CollectionItemControllerTest	1	0	0	0	100%	0.623
CSIControllerTest	1	0	0	0	100%	1.058
GameControllerTest	1	0	0	0	100%	0.789
GameReltnControllerTest	1	0	0	0	100%	0.274
MMCControllerTest	1	0	0	0	100%	2.829
UserControllerTest	1	0	0	0	100%	0.658

ac-games-restservice-spring
0.2.0-SNAPSHOT
Clover Coverage Report

Dashboard
Coverage Reports
Coverage (Aggregate)
Test Code (Aggregate)
Test Results

Application Packages

com.ac.games.rest (0%)
com.ac.games.rest.controller (41.4%)
com.ac.games.rest.data (58.7%)
com.ac.games.rest.filter (0%)

Classes Tests Results

Test Case	% Success
BGGControllerTest	(100%)
CSIControllerTest	(100%)
CollectionControllerTest	(100%)
CollectionItemControllerTest	(100%)
GameControllerTest	(100%)
GameReltnControllerTest	(100%)
MMCControllerTest	(100%)
UserControllerTest	(100%)

Clover Test Report - ac-games-restservice-spring 0.2.0-SNAPSHOT
Coverage timestamp: Wed Mar 18 2015 14:13:00 CDT

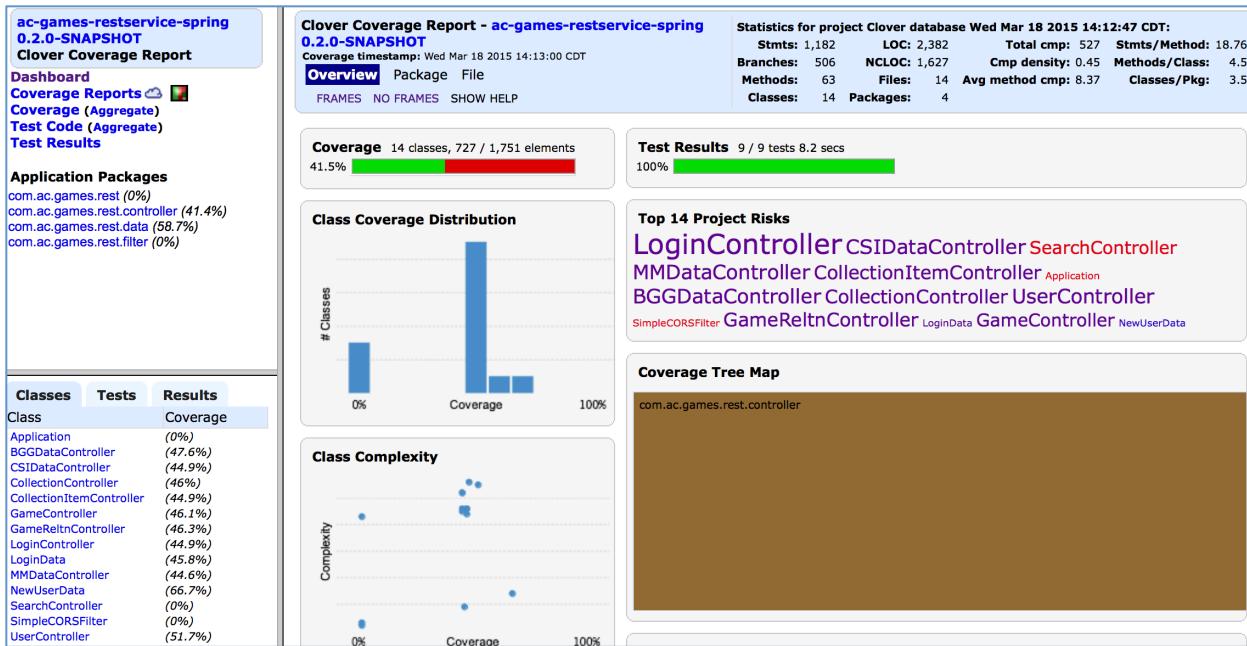
App Test Results Clouds

Overview Package File Test
FRAMES NO FRAMES SHOW HELP

Project Tests Fail Error Time % Pass
Clover database Wed Mar 18 2015 14:12:47 CDT 9 0 0 8.202 100% 

Packages Tests Fail Error Time(secs) % Pass
com.ac.games.rest.test 9 0 0 8.202 100% 

Report generated by Clover Code Coverage v3.3.0 Wed Mar 18 2015 14:13:42 CDT. Clover: Commercial License registered to Cerner Corporation.



Surefire and Clover Reports for this project are stored under <https://github.com/apshaiTerp/cs-5551-documentation/tree/master/maven%20sites/ac-games-restservice-spring/ac-games-restservice-spring-0.2.0-SNAPSHOT%20site>.

VI. Deployment

The ScrumDo link to this iteration can be found here:

<https://www.scrumdo.com/projects/project/terrapin-collection-manager/iteration/121582>

As discussed previously, there are 6 stories that were completed this iteration, and 6 more that are in-flight that will be rolled into the next iteration.

The final end-states for my projects has been included in the GitHub documentation site, but this is the list of the final java project end-states:

[ac-games-pojos-0.2.0-SNAPSHOT.jar](#)
[ac-games-db-0.2.0-SNAPSHOT.jar](#)
[ac-games-db-mongo-0.2.0-SNAPSHOT.jar](#)
[ac-games-restservice-spring-0.2.0-SNAPSHOT.jar](#)
[ac-games-agent-0.2.0-SNAPSHOT.jar](#)

Because I have chosen to implement my project in Java, I needed to find an alternate form of deploying my REST service and other application elements so they can be publically accessible. To this end, I used a spare computer at home to install an Ubuntu desktop, from which I am running my own MongoDB Instance and Tomcat Server. The Tomcat server hosts both my REST service as well as the web site being developed for this project.

The root URL for my service is <http://107.188.249.238/ac-games-restservice-spring-0.2.0-SNAPSHOT/>.

The URL for my site is <http://107.188.249.238/tcm/>

All screen caps for the website generated above were taken using the live site deployment.

VII. Project Management

Because I am a team of one, I can attest that all work completed on this project is my own. While keeping track of the time spent has been difficult (I don't track time spent, especially now that I can't track it in ScrumDo), I would estimate that I spend about 20-25 hours weekly working on this project. I believe my code output is indicative of the time commitment I have made to work on this project.

While I remain optimistic at this point that I will complete all of the desired features and stories by the end of this project, I have had concerns that I might have overreached in my ambition. I will admit it is proving difficult to maintain this level of effort while working full time and having a second class. Part of this fear is largely due to the fact that until the last week, I have been working exclusively on framework components. This is an effort I believe will be valuable in the long run, but has at times made it difficult to gauge how much progress I was making on the application as a whole.

I believe the end of the third iteration should give a better measurement of how close I will be towards satisfying my primary goals with this application, as I hope to devote much more time to the client-side development work.

VIII. References

- [1] – <https://github.com/apshaiTerp/cs-5551-documentation/blob/master/CS551-Project-Proposal.pdf>
- [2] – <https://github.com/apshaiTerp/cs-5551-documentation/blob/master/CS%205551%20Project%20Plan.pdf>
- [3] – <https://github.com/apshaiTerp/cs-5551-documentation/blob/master/CS%205551%20First%20Increment%20Report.pdf>