CSCI E-97 Software Design: Principles, Models, and Patterns – Fall 2014
Assignment #4 – Results
Roland Galibert
November 20, 2014

Following is a summary of my overall experience in assignment #4, designing and implementing the modular Authentication Service, intended for use with the SquareDesk application as well as other applications.

Test driver notes:

In addition to the TestDriver which runs correctly according to the assignment requirements, I provided several other tests which can be run as follows:

java -cp . cscie97.asn4.test.LoginLogoutTest // Tests AuthService login/logout functionality and exceptions

java -cp . cscie97.asn4.test.OfficeSpaceTestDriver // Original test driver from assignment 2, now including authentication

java -cp . cscie97.asn4.test.SearchEngineTest // Original test driver from assignment 3, now including authentication

java -cp . cscie97.asn4.test.SchedulingServiceTest // Original test driver from assignment 3, now including authentication

Output for these tests have also been included in the testOutput/ folder.

Notes regarding TestDriver output

I added registerWithAuthService() methods to each of the OfficeSpaceProviderServiceAPI, RenterAPI and (SquareDesk) UserAPI classes from assignments 2/3 which, when the singleton API classes are first instantiated, creates/registers those services with the authentication service, registers the respective restricted access methods and creates appropriate roles.

Since my AuthService bootstrap method creates a number of authentication service objects (a super admin profile and user as well as a user role with limited permssions) and also registers the restricted methods of the AuthService itself, all these objects will be included in the AuthService object inventory generated in the TestDriver, along with the AuthService objects in authentication.csv. The only modification I had to make to the test .csv file was to change the permission “create_user” to “create_user_permission” (since this restricted method permission ID appears in my boostrap method).

I understand that creating auth service methods as well as SquareDesk method are part of the test, but I couldn’t think of a way to let the system “build itself” on the basis of just the test data without circumventing the requirement that the AuthService methods are themselves restricted (though of course this is what I did in my bootstrap method).

Design review

My design review went very well. My original partner (I suspect because of reasons of work and time) suggested that in addition to him, I ask one of the instructors to review my design document, and I asked Sytze to review it. This was very helpful, since because of his greater experience I received a lot of good pointers.

Sytze's main suggestion in terms of actual design was to create an authentication service interface in addition to the actual implementation. On the basis of what he said, I also realized I hadn't given enough thought to the actual implementation of the authentication service visitor interface and the authentication service visitor element (visitable) interface/implementation, so I thought that design through a little more.

Sytze also gave me a lot of good pointers in terms of actual presentation of the design document, suggesting I give a "10,000 foot" overview of the design, including a better overview of all three SquareDesk components (Provider/Renter/Authentication), adding more detail in my class descriptions to point out what requirements are met, and adding more explanation to my diagrams. Unfortunately, due to time, I may not be able to implement all of these design document suggestions.

Design changes

Due to time, I wasn't unable to read the input from the discussion board nor again review the relevant material in the lectures until after I actually created my initial design document, so my actual authentication service implementation has quite a few changes from my original design.

The biggest things I took away from the discussion board was the concept of the authentication service as a standalone service which could support any application, not just the SquareDesk solution, and the need to add bootstrap functionality. As I mentioned in the note above, my bootstrap method included both creation of a bootstrap user as well as registration of the authentication service methods themselves as restricted methods.

Then I also made the following other changes from my original design:

- Credentials – allowed a set of credentials (instead of a single credential) to be assigned to a single user, revised hashing function (please note much of my implementation in this regard is based on the article Professor Giesecke posted, "How to generate secure password hash : MD5, SHA, PBKDF2, BCrypt examples" by Lokesh Gupta on the "How To Do It In Java" website).
- Access token – Again, I changed my initial implementation concept after reviewing the discussion board and added a TIMEOUT_PERIOD constant and lastAccessTime property.
- Visitor – Added AuthServiceVisitorElement (i.e. a visitable) interface which is implemented by the Service, Permission, Role and User classes
- IDs vs. UUIDs – Probably because of the last two assignments (and rushing due to time) my initial design used UUIDs for services, roles, permissions and users instead of String IDs, so my final implementation better reflects the actual requirements.

Usefulness of design document

Creating the design document was definitely useful, and I feel I'm getting much better at what is required in terms of detail and modularity, both in the design document and in the design itself (e.g. how complicated/simple to make an class, where to throw exceptions, etc.). I can see how the

recommended structure of the design document is useful, it implements a somewhat “top-down” approach, and the different diagrams (especially the use case, sequence and activity diagrams) give you a general idea of the system so that the class diagram and class dictionary can be designed better and more quickly. As always, I wish I had had more time to devote to the design document process.