Results Document

Date: 11/20/2019

Author: Tofik Mussa

Reviewer(s): Trisha Singh, Peter Chen

The use of design patterns especially the Visitor pattern simplified a complex project into decomposable and easy understandable units. This module was also the last of the Store 24X7 system and that highlights the importance of breaking up a problem into pieces that belong together then defining clear boundaries among the pieces hence loose coupling and high cohesion. The authentication service is also a standalone and generic module which adds to its reusability by future systems. By anticipating future changes, the design relies on interfaces in lieu of concrete implementations. The composite pattern also produced a tree like structure which helps clients to treat leaf and composite objects in the same regard.

The design was done in multiple iterations and there were some ambiguous requirements at the beginning. I did a few diagrams on a whiteboard and Astah that I eventually had to throw away before connecting the dots. When I got to a point where I couldn't improve the design anymore except for some minor fixes, I started implementation. I am looking forward to a feedback for what could have been done better. There were some details that were unclear until implementation began like when looking for permissions in the CheckAccessVisitor class. It was not obvious how backtracking and recursion worked in the entitlement tree until fiddling with the code and getting it right. The assignments were of increasing complexity and even though design has been getting easier, the scope and intricacy of the problem is also growing.

Peer Review

The peer review was helpful specially to talk out loud about the requirements. I discovered finer implementation details that I didn't realize initially. I had a back and forth discussion especially Trisha Singh

Trisha's comment on my design

1. I see the generateToken function takes in the credential, is the username an attribute in Credential?

2. I see all components of Authentication service including User, AuthToken, Resource extend Visitable. I don't think that's necessary although it's certainly not incorrect.

3. It’s nice that you’ve encapsulated the different types of credentials into different subclasses.

In general, there is no major flaw with your design. It will certainly work.

My response to Trisha

1. **I see the generateToken function takes in the credential, is the username an attribute in Credential?**

I have updated the redundancy if you take a look at the credentials now. Credentials can be either username/password combo, face print or a voice print.

2. **I see all components of Authentication service including User, AuthToken, Resource extend Visitable. I don’t think that’s necessary although it’s certainly not incorrect.**

I am having them as vistables even though you can name them anything. It is a convenience for the inventory visitor. Since they all implement accept() method, I extracted that method into an interface.

3. **It’s nice that you’ve encapsulated the different types of credentials into different subclasses.**

Thank you. Since any of the 3 credential types can be used interchangeably, i found it as a good opportunity to exercise polymorphism

I gave structuring credentials and its relationship with a user a second thought after this discussion.

My suggestion to Trisha

I see in your diagram that resources are managed by the AuthenticationService class. I have not thought about doing that. My question for you is if you have thought about how to implement the hasPermission method. Don't we need to pass in the resource as well to check if the user has restricted access to some resources? And how are you trying to incorporate the visitor pattern since it is a requirement?

Trisha's response

> Regarding your questions about my design:
>
> 1. I missed including the Visitor pattern in the design.
> 2. About permission and resource - I see you pass them as two separate arguments and that's a great idea. I was thinking of passing those two as one string as I was imagining making use of the resourceRole class to check for permissions.
> I need to put some more thought into this.

My suggestion to Peter

> In my opinion having the AuthService class manage AuthToken has some benefits. We can perform the first hand validation before performing any of the subsequent steps to check for say permissions. That may save us a lot of trouble if the authToken was invalid to begin with. It also helps with navigation from having to think through my design.
>
> A special type of Role called ResourceRole is also a requirement when a user has access to a certain resource rather than a global access to all resources.
>
> Your design captures most of the requirements but I believe it is still under construction. Please let me know your thoughts.

Overall, the peer review has been specially useful to brainstorm and wrap our head around requirements.

Commands to run the program

#Compile

javac com/cscie97/store/ledger/*.java com/cscie97/store/model/*.java com/cscie97/store/controller/*.java com/cscie97/store/authentication/*.java com/cscie97/store/test/*.java

#Run a happy path test script

java -cp . com.cscie97.store.test.TestDriver store.script

#Validate how exceptions are handled

java -cp . com.cscie97.store.test.TestDriver store-exception.script