Final Assignment- LCM Ltd. Company Grails Application

Before starting explaining my submission, I would like to say that *I would like to be considered for production usage* cause I think it's a very good opportunity to have a first contact with the business world and real projects.

In order to explain how I have made this assignment I would like to follow the steps I have followed doing it.

1. Install SpringSecurity Plugin

First of all, we have to install the SpringSecurity Plugin in our project following the instructions that we have learned in class. With the s2-quickstart command we will have a LoginController, LogoutController, User, Role and UserRole Domain classes. Using the plugin we would be able to login into the application with the users and roles that we will define later.

2. Add company field to user domain class

In order to have a company field for each user of our application we would need to add a String company to the default values in the User domain class.

3. Boostrap.groovy

Boostrap.groovy that can be founded in the config folder in our project directory enables us to create the roles, the companies and the users of our application.

I would like to attach the code here to explain it better.

```
def adminRole = new Role(authority:'ROLE_ADMIN').save(flush:true)
def userRole = new Role(authority:'ROLE_USER').save(flush:true)

def LCM = new Company (name:'LCM').save(flush:true)

def testCompany = new Company (name:'testCompany').save(flush:true)


def admin = new User (username:'admin', password:'admin',
company:'LCM', enabled:true) admin.save(flush:true)

def sampleUser = new User (username:'sampleUser',
password:'sampleUser', company:'testCompany', enabled:true)
sampleUser.save(flush:true)

UserRole.create(admin, adminRole, true)
UserRole.create(sampleUser, userRole, true)
```

As we can see from the code, we have created an adminRole and a UserRole. Then, we have defined two companies: LCM and testCompany. We also have created two users: admin that has LCM company and sampleUser that has testCompany. Finally, we have matched the roles with the users using the UserRole. Create function.

4. Apply ADMIN_ROLE privileges to all actions in Location Controller and Company Controller

The next step to build our application is to delete the scaffolded controllers that we built in the last assignment and generate them. We have generated Location and Company controllers and, after that, we have used the security plugin adding the code:

```
@Secured(['ROLE_ADMIN'])
```

This code enables us to restrict the pages for the company and location to our admin users giving us a message if the user is not the admin one.

5. Login and Logout

To login and logout as the desired user in our application we have included in the main layout the links to the Springsecurity login and logout controllers that enables us to login if we are not logged in, and in case we are, give a link to the logout one. Moreover, if we are logged in the application will display the user as we are logged in. We have included some CSS here to display them better and easier.

6. ReportController

As we need to save the report as a Report Domain Object we need to use the controller Report and the Domain Report so, we have generated a ReportController and used and modified the default methods to do all the Report functionality.

○ Action save

First of all, we have to know that the save action is going to save the Report Domain object when the form is filled in and the object is created and so, the file is uploaded. For this reason we have to restrict this action to the admin users as they are the only ones that can upload a report and save it.
This method has three parts:

- The first one, that I attach the code here, uploads the report to our grails project directory. I have created a folder inside it to the uploadedfiles and using the function getOriginalFilename it will save the report with the same name as the original uploaded one.

```
def f = request.getFile('file')
       def namefile = f.getOriginalFilename()

       if (f.empty) {
              flash.message = 'file cannot be empty'
              render(view: 'create')
              return
       }//if
       f.transferTo(new File("../GrailsProjectDirectory/web-
       app/uploadedfiles/${namefile}"))

       params.filename = namefile
```

- The second part is needed to complete the params that the Report Domain Object needs to be created and saved. So, using params we are going to include the filename and the publicationdate that we need.
- The last part is to save the Report domain object and once saved, it would show us the params of the Report domain object created using the action show.

```
     def reportInstance = new Report(params)
        if (!reportInstance.save(flush: true)) {
              render(view: "create", model: [reportInstance:
              reportInstance])
            return
        }

        flash.message = message(code: 'default.created.message', args:
[message(code: 'report.label', default: 'Report'),
reportInstance.id])
        redirect(action: "show", id: reportInstance.id)
```

View create

This view is the one that we press the button to upload a report. This view is using the information in the form.gsp view. When we press the button “upload” that we can see from the application we are getting into the save action so, this view, will also be exclusive for admin users.

Action list – Graduate extension

This action allows us to see the reports that we have uploaded. Both roles of the application can see the uploaded reports, so, we are not going to restrict it writing any @Secured code before the action in our ReportController.

In this action, we have to include the graduate extension to see only the reports for our company, and grouped by location and ordered by its publication date, from the most recent to the least.

I would like to attach the whole code for this action here:

```
def list(Integer max) {
```

I have done all this code trying to do the graduate extension but finally, it took me a lot of errors and running out of time, I have preferred to give the code that works (subtract a year from the current date and take the company for the logged user) and leave the rest commented. However, I would like to explain how I was trying to do this.

```
        def now = new Date()
        def yearFromNow = now.minus(365)

        def user = springSecurityService.getCurrentUser()

        def companyUser= user.getCompany()
```

I was taking the locations for the company related to the logged user. Then, I was taking all the reports for this company and location. And finally, defining a criteria that takes from this report list, the reports that was uploaded between now and a year ago, I was trying to group them by locations and then, ordering by the publication date.

```
        //def locations = Location.findAllByCompany(String.companyUser)
        //def reportslocation = Report.findAllByLocation(locations)

        //def criteria = Report.createCriteria()

        //if(!reportslocation.isEmpty()){

        //reportslocation = criteria {
               //between("publicationdate", yearFromNow, now)

               //and {
                     //groupProperty("location")
               //}
               //order("publicationdate", "desc")
        //}//criteria

               //render (action: 'list')

        //}else{

        //println "There are no currently reports available"

        //}
    params.max = Math.min(max ?: 10, 100)
           [reportInstanceList: Report.list(params), reportInstanceTotal:
  Report.count()]
}
```

7. CSS

To make our application beautiful, we have included some CSS in the CSS folder using Bootstrap code, buttons and navigation bar, so, we can move easily to the pages of our application.

Here is a picture of the final application and the final navigation bar: