DESIGN PAPER

Justin Francis

A20314408

A community library needs an inventory management system. The aim is to describe how the library inventory management system can be implemented using this design paper.

Library Inventory Management System

Introduction:

A community library needed an inventory management system. The library inventory management system will have different user roles for the different actions that they can perform. Some of the actions that these users can perform will be to add a new inventory item, mark old ones not for loan as retired, loan the book, find the most popular author, sort inventory items into different sections. For this, based on the ease of the implementation, it was decided to use grails

Project Requirements:

For ease and simplicity in creating this simple system, grails will be the best way to implement this. Grails / Groovy Tool Suite will be a simple and easy IDE to implement the design. The library inventory management system can have certain inventory items like books, blu-ray disks and magazine. The library also keeps track of the most popular authors by noting down how many books of a particular author are on loan. All inventory items are organized into library sections. The various user roles defined in the application and their actions are:

Library Manager

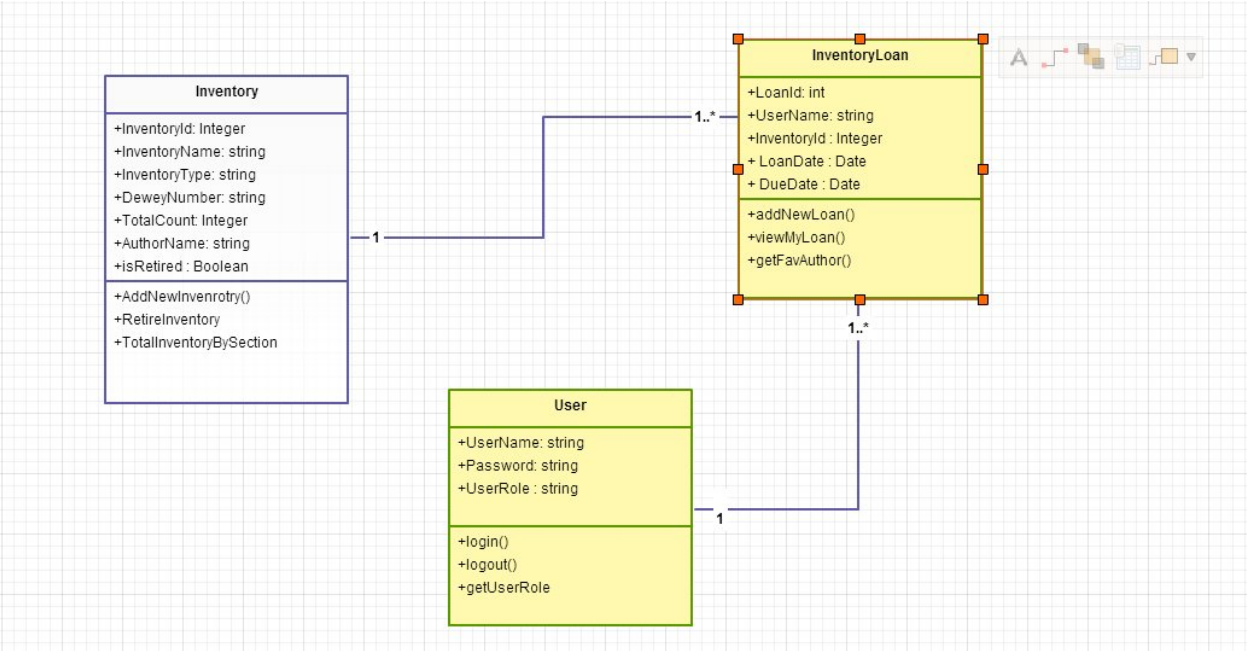
- A library manager can add new inventory items.
- A library manager can mark old inventory items not for loan when they are retired.
- Manage library sections by viewing the total number of books available in each section.

Library Desk Operators

- A library desk operator can scan inventory barcodes and record the loan date, due date and the user information who loaned the item.

Regular Users

- Regular users can loan a book.
- View loaned items and note down when they are due.



MVC Elements And Relationships

Model

Inventory: The inventory model will consist of the following fields:-

- Integer inventoryID
- String inventoryName
- String inventoryType
- String dewyNumber
- Integer totalCountofInventory
- String authorName
- Boolean isRetired
- String inventorySection

An inventory will consist of a unique identifier (primary key) which will be the inventoryID. This will be a primary key in the database. An inventory Name can either the book name or blu-ray disk name or magazine name or any other inventory item that maybe added later on. The inventoryType will be a book, a magazine or a blu—ray disk. Again depending on the inventories, a new type maybe added and the corresponding type name can be added here. The DewyNumber field will store the dewynumber of each inventory item. This will be used to segregate inventory items based on different sections. The totalcountofInventory field will consist of the total count of the particular inventory. A magazine can contain more than one count of the same item. Any inventory can contain an author. Here I am assuming that an inventory item can contain only one author at all times. The inventory item can also be retired and hence the isRetired field is used to check if the item is retired or is still active. The inventory section is calculated on the dweyNumber using its first three characters and thereby making the corresponding entry into the inventorySection.

User: The user model will consist of the following fields:-

- String username
- String password
- String userRole

A user can be identified by his username and password. These credentials must be used to login to the application. The username will be unique and hence will also be the primary key. A user can be of different types of roles. A user can be a library manager, library desk operator or normal user. Based on the different roles, the user can perform different operations. A normal user can loan inventory items and view when they are due. A library manager can add new inventory items in the database and can also retire a book so that it may not be loaned again. Library desk operators can scan the inventory barcodes and record the loan data, due date and the user who loaned it.

InventoryLoan: The inventory loan model will consist of the following fields:-

- Integer loanID
- User user
- Inventory inventory
- Date loanDate
- Date dueDate

This model will contain all information with regards to a loaned inventory item. Each item loaned will have a loanID which will also be the primary key. It will contain the user information from the User model and hence username will be the foreign key in the database. It also needs to note down the inventory that is being loaned and hence inventoryID from the Inventory model can be used here. It will be a foreign key in the database design. A user can have multiple loan items and hence there is a one to-many-relationship between user and inventoryloan. An inventory can have multiple loan items and hence there exists a one-to-many relationship between inventory and inventoryloan.

View

The following views will be essential to display all required views in the application:

Index view: The index page will be the home page and contain a login form to login into the inventory management system. If a user has been authenticated then he gets redirected to his respective login page based on the user role. The user will now get to view default views for a type of user role assigned with their login. A library manager view will contain the following options once he logins in, add a new inventory item, retire an inventory item and view total number of books in each section.
AddInventory View: The view is only available for library managers. It will be a form containing inventory item fields where the manager can use this form to create a new inventory item to be made available to the user for loan.
RetireInventory View: This view is only available to library managers and is used to retire an inventory item. The existing list of items can be displayed and the user can mark it as retired so that it is no longer available to loan.
TotalInventoryInSection view: In this view the library manager can view the total books in a section.
LoanNewInventoryItem view : This section is available to library desk operators. A library desk operator view will have the option to scan an inventory item and reads its barcode. Once the corresponding item information is displayed, the desk operator can fill a form containing information about the user and the loan date and due date information.
LoanedItems view: A regular user view will contain a list of all loaned items sorted by type and will show when it is due.

Controller

The following list of controllers and its methods must be used to implement this project:
User Controller: This controller will implement login and logout methods to authenticate a session. The user role will also be determined and based on the user role a user will be redirected to its corresponding page after logging in. On unsuccessful authentication, the user gets redirected back to the index page displaying an error message.

Inventory Controller: This controller will have the inventory methods implemented in it. It will have the method to add a new inventory called addNewInventory. It will also have a method to retire an inventory called retireInventory by updating the isRetire field of the application and item. A method to calculate the total inventories called totalInventoryBySection in a section by

using the dewy number field of each item by iterating through the entire item list and making separate list of sections based on every dewey decimal section.

Loan Controller: This controller will contain a method called addNewLoan to add a new loan item for a particular user. The user information will be entered by the library desk operator. This controller will also contain a method called viewMyLoans to view the loans for a particular user. A user will login to the application and based on the session its loans will be calculated. This section will also contain a calculateFavAuthor method which will find the count of all loaned items for a particular author and get the author with the maximum count. Here getting the loaned items for a particular author can be done by querying the database using select count(author) from tablename groupby author.

Application Code Layout

The following application code layout can be implemented as:

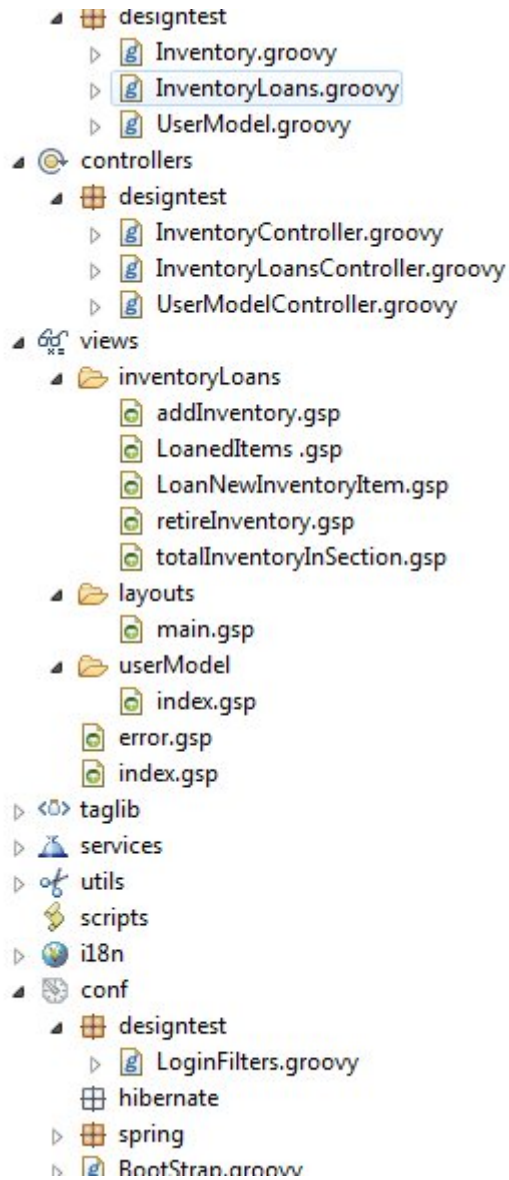
Domains will be

- Inventory
- User
- InventoryLoan

Controllers will be

- InventoryController
- InventoryLoanController
- UserController

Views will be as shown below in the diagram.

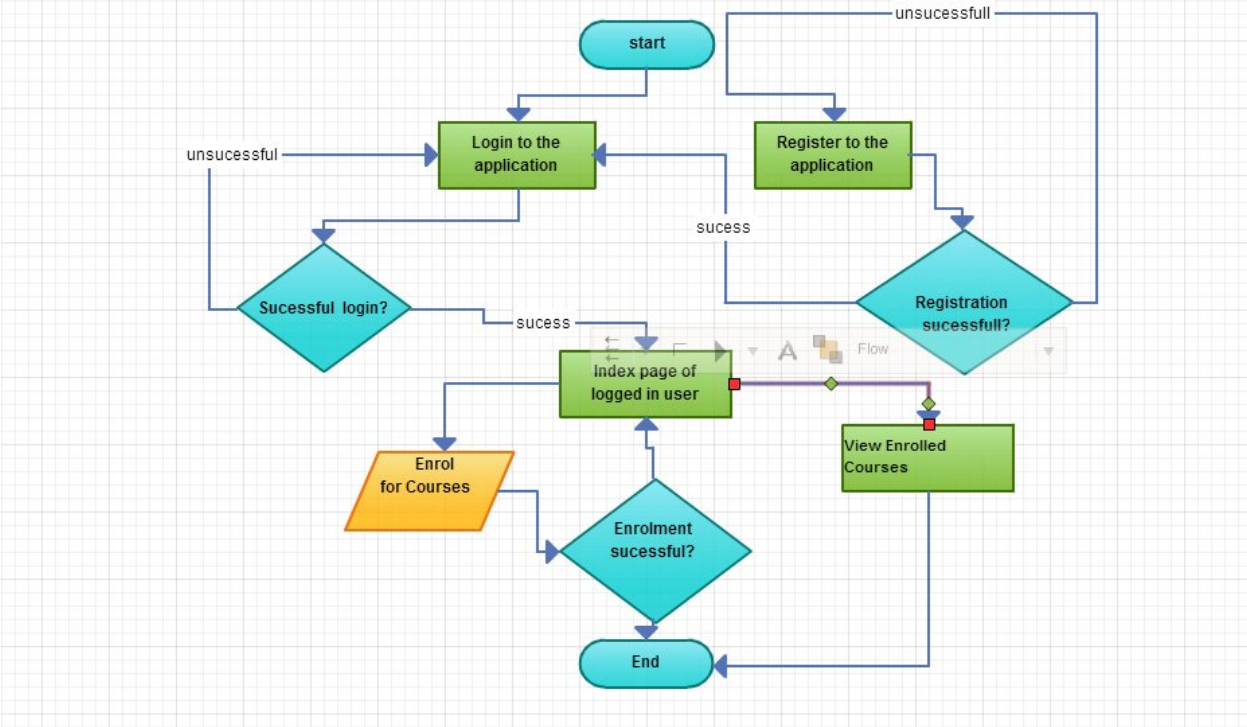


Filter must be implemented to check if a user is logged in before the control goes to any other controller so that a user not validated cannot enter the page.

Major visual layout considerations

The main layout will contain the following common links:

- Site title
- Home button to direct the user to his sessions home page or redirect to login page If user is not logged in.
- Logout button if user is logged in
- If library manager is logged in then it will show links to add new inventory, retire inventory and manage library sections.
- If a library desk operator is logged in then there will be links to add a new loan item.
- If a regular user is logged in, there will be links to view loaned items.



Security and access control

A user will be validated to check and display appropriate pages and layouts to the user. A user if not authenticated will not be able to navigate to any of the inner pages as filters. Since there is a userRole field, on validating a user, there is a check made to check the user role so that users with specific role can only view specific pages. A check can be made in the controller to route the user to the appropriate page.

Grails automatically escapes SQL characters to prevent SQL injection. Using positional parameters to pass parameters helps in avoiding SQL injection. HTML/URL injection is easily handled with the codecs supplied by Grails.

Deployment Strategy

Use the following steps can be used to deploy the grails application into the production setup.

1. If necessary, install the MySQL-JDBC driver (currently mysql-connector-java-5.1.25-bin.jar) into the lib directory. Copy it from another Grails project or get it here:
2. Set the information in the BuildConfig.groovy config file:

   dependencies {

   runtime 'mysql:mysql-connector-java:version_No'

   }

   The version information must match the driver file version.
3. Set the information in the DataSource.groovy config file using values for url, username, and password which are appropriate to your installation (using the production database):

```
production {
    dataSource {
            db = “update”
            driverClassName = "com.mysql.jdbc.Driver"
                url = "jdbc:mysql://localhost/prod"
                username = “”
                 password = ""
    }
}
```

4. Now we have to make changes in the Bootstrap.groovy

   ```
   if (GrailsUtil.environment == "production") return;
   ```

5. Build the WAR file. Choose Run->Build Project from the menu. The WAR file will have the project name with the version info. Move the war file into the Tomcat sub-folder. Restart Tomcat and run it.

Additional assumptions and questions you would need answered to continue

- Here we are assuming that an inventory can have a single author.
- A user can have one of the three types of roles: Library manager, Library desk operator, regular user.