Java-FLP parallel project

(Project along with regular learning sessions)

FMS – Film Management System

INDEX **Page Number**

APPROACH AND EXECUTION PROCESS

Learning project for FLP associate in parallel to their training - Basically participant will work on this after daily's regular session hours, and on Saturdays. So time for FMS work will be evening 5:30 to 7.

Initially, two full days will be dedicated for FMS; to understand the project; to decide the package structure, nomenclature, data model (domain objects) and start the work. After two days, FLP participant will keep working on it till the final project comes.

Project tile - FMS (Film Management System)

APPROACH - Associate will develop the application in 4 phases, in all 4 phases complete requirement should be achieved.

Phases -

- **First**: immediately after completion of Core Java sessions (after collection API) – Associate will develop application using core Java only.
- **Second**: after RDBMS (SQL) and JDBC session – use of real DB for the application
- **Third**: after the Servlet & JSP session – revamping of the application using the JSP and web technologies like JS, HTML.

It is desirable that in all phase – complete functionality (create, modify, remove, view and search) should be achieved; but it should not be developed every time. It should follow a good package structuring (presentation, business and persistent layer), so that in phase 2 (when JDBC and database will be introduce) only persistent layer should be change. Use of proper interfaces and nomenclature is desirable. Similarly in case of conversion from plain Java application to web base application, only presentation layer should change and no change should be required on business (service) or persistent layer.

Purpose of this exercise is to understand benefit of proper MVC structure, layer based design, reusability and ease of development (realization of requirement) with advancement of technology.

In other word they will EXPERIENCE the power of technology and its real usage.

ABOUT FMS

FMS is an application, which helps to manage an actor details like – personal and professional info of actor. Personal info like First Name, Last Name etc. And professional info like –Films, Film Category, Film Rating etc.

FMS aims to manage complete life cycle of film viz **add** (create a film – where all info will be captured for the first time), **edit** (modify film info) – like roles, films, etc. **Remove** (delete film) film data from system. Along with add, edit and remove – system should display all actor summaries (**View**) and must support **search** facility.

The system will first develop using core java only – where film data will be store as a Collection (List, set or Map). For user interaction, system will use Scanner API.

Later on data will be store on My SQL database; system will use JDBC for the same.

In Phase three –FMS will become web based application, following MVC design pattern. Here the application will be develop in JSP – Servlet.

Macro level Operations/offerings:

1. Create Film (capture film info)
2. Modify Film info
3. Remove Film
4. View all Film (Film summary)
5. Search Film by name
6. Search Film by Actor
7. Search Film by Category
8. Search Film by Language
9. Search Film by Rating

MODULE LIST and MODULE DETAILS

CREATE FILM

Following info need to capture

- film title
- film description
- release year
- rental duration
- rental rate
- length
- replacement cost
- rating

SEARCH FILM

User should be able to search a film by film title, release year, rating User may use any one, two or all the three parameters to search a film.

MODIFY FILM

Search (film title, release year, rating) film and modify the info of film. System should show existing data/info of film and should support modify of one, more or all info.

REMOVE FILM

Search (film title, release year, rating) and remove the film. System should ask for confirmation and on confirmation the data will be removed.

FILM SUMMARY (VIEW)

System should show (display) film list in a tabular format (one row for each film, and columns for film info). It is not required to show all the data of film in table; only important info likes – film title, release year, rating. project is needs to show in table.

Film Report

1. Search Film by name
2. Search Film by Actor
3. Search Film by Category
4. Search Film by Language

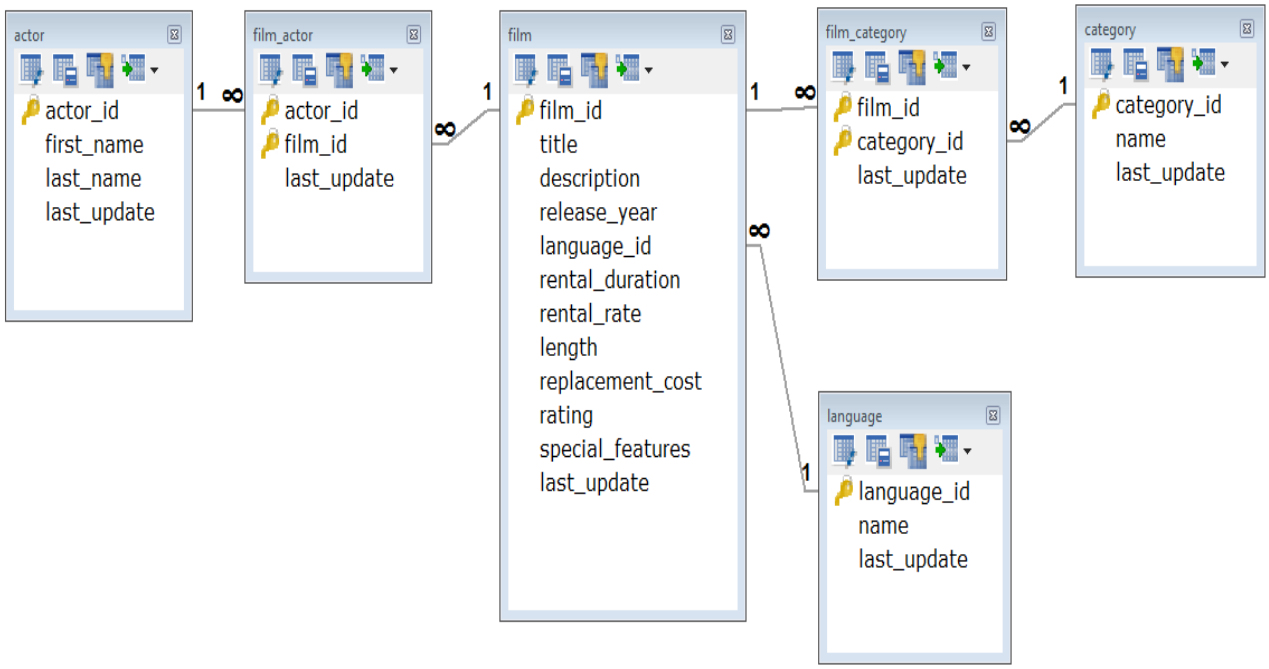
5. Search Film by Rating

Constrains

- Proper validation is required (specially film title, release year, rating) System must check uniqueness on film title, release year, rating
- System must show appropriate massage on all activity (whether activity is successful or failure)
- User must have proper menu to select the activity (create, modify, search, view, remove) that user want to perform.

DATA BASE

Tables



PACKAGES AND THEIR REQUIREMENT

In Phase 1 (completely developed in core Java)

Layer	Package Name	Class/interfaces	Purpose
View	com.flp.fms.view	BootClass Will have two method –	Main method start point of application; System will show menu to user and

		p.s.v.main (starting point of application – it will call other method) menuSelection	user will select menu (using **Scanner API**) menuSelection method – will use scanner to interact with user and switch to direct the user instruction to appropriate method of UserInteraction class
		UserInteraction (with 10 method AddFilm, ModifyFilm, RemoveFilm, SearchFilm, getAllFilm AddActor, ModifyActor, RemoveActor, SearchActor, getAllActor)	User will enter/view the data based on activity type (using **Scanner API**). For Create and Modify - System will capture date (validate it) and create a HashMap and pass it to next Layer For Search, View and Remove – capture the Film id on which activity needs to perform and pass it to next layer
Service	com.flp.fms.service	Interface - IFilmService (with 10 method AddFilm, ModifyFilm, RemoveFilm, SearchFilm, getAllFilm Interface- IActorService AddActor, ModifyActor, RemoveActor, SearchActor, getAllActor) Classes – FilmServiceImpl ActorServiceImpl Which implements the interface	Receive data from UserInteraction class (view layer). For create – instantiate a new Film and Actor object; populate it and pass it to next layer (DAO) - here system must check Film and Actor uniqueness (film title, release year, rating, Actor Name, Film Name) and must avoid duplicity of data For other activity – act appropriately on Film and Actor object and pass it to DAO

Dao	com.flp.fms.dao	Interface – IFilmDao (with 5 method AddFilm, UpdateFilm, RemoveFilm, SearchFilm, getAllFilm Interface- IActorDao AddActor, UpdateActor, RemoveActor, SearchActor, getAllActor) Classes – FilmDaoImplForList ActorDaoImplForList Which implements the interface	System will receive data from service layer and perform appropriate action In case of create – add new Film object into Array List. In case of remove – remove the selected Film object from the list. In case of modify – replace the selected Film object,.
N/A	com.flp.ems.domain	This package will have following Classes Film Role Department Project	POJO (bean) classes – having required properties and getter and setter methods. For Film class – it will be required to override equals and hashCode method corresponds to film title, release year, rating properties. To avoid duplicity of data.
N/A	com.flp.ems.util	This package is to keep all utility classes – for now there will be only one class Validate	Validate class will have static method with return type boolean, to validate – film title, release year, rating, actor name etc. User regEx to validate date

In Phase 2 (introduction of JDBC and My-SQL database)

Changes

Only change will be – Now IFilmDao interface will be implemented by class FilmDaoImplForDB (instead of FilmDaoImplForList).

Since, now system will be storing data into database, in place of array list.

In Phase 3 (Convert the application as web application – using Servlet and AngularJS)

Changes

- No change in business and persistent layer – but view layer will be changed completely
- No need of BootClass – as now web.xml will be the starting point of application and menu will be available on the index page (welcome page (jsp) of application)
- No need of UserInteraction class – as now Servlet will capture data from user (request.getparameter("parameterName"). The Servlet class will send data to next layer in a similar way (as userInteraction class), i.e. in case of For Create and Modify - System will capture date (validate it) and create a HashMap and pass it to next Layer
- Also now the Validate class will not be in use; all fields will be validated at client side using JavaScript

TECHNOLOGY AND ARCHITECTURE

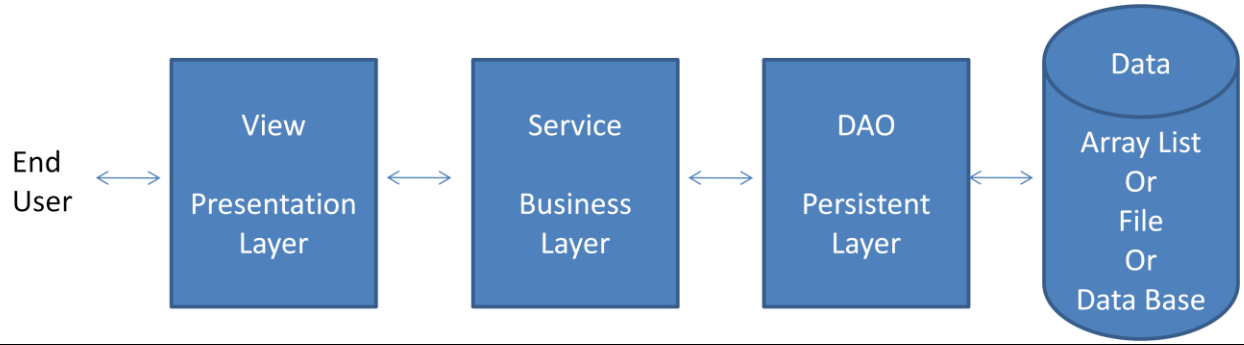
TECHNICAL SPECIFICATIONS - Software Requirement:

- Spring Source Framework 2.5.2
- Tomcat Server 6
- My SQL 5.0 (From phase 2)
- JDK 1.6

PHASE WISE TECHNOLOGY FOCUS -

Phase		
1	Only Core Java	Special attention on – Collection API Scanner Calender RegEx Proper package structuring
2	Database	JDBC My SQL
3	Web based	HTML (front end) Java Script (Specially for validation) JSP Servlet
4	Spring MVC based	Spring MVC IoC Spring JDBC

ARCHITECTURE – Layers


End
User
View
Presentation
Layer
Service
Business
Layer
DAO
Persistent
Layer
Data
Array List
Or
File
Or
Data Base