UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

SocialMovies Mini-project Web Engineering 2015

Nadia Aebsicher
Simon Brulhart
Jocelyn Thode

December 12, 2015

Mini-Project Report

Department of Computer Science • University of Fribourg • Boulevard de Pérolles 90 •
1700 Fribourg • Switzerland

http://diuf.unifr.ch

Contents

1 Introduction

We used Ruby on Rails and RDF to create this project. The resulting web application lets you search movies, favorite and share them with your friends as well as getting personalized recommendations based on what you liked. You can also add movies to different lists such as 'Watch it later' or 'watched'.

In the next chapter, we will present our development environment as well as the different technologies used for the project. In chapter 3, we'll present the Ruby on Rails framework and explain how the MVC architecture works.

In chapter 4, we will present which RDF database we used and why RDF is useful.

In chapter 5, we will talk about our application, how it works and some interesting features.

Chapter 6, will be about the installation and user guides. Finally we will conclude the report and reflect on what was achieved for this mini-project.

2 Development Platform and Environment

We used the following environments to develop our application :

- **Operating Systems**: Mac OS X 10.11, Archlinux
- **Web Browsers**: Safari 9.0.1, Chromium 47.0, Firefox 42.0
- **Ruby 2.2.3** is the programming language used by Rails
- **Rails 4.2.4** is an MVC framework written in Ruby. It lets you develop web applications using the DRY (Don't Repeat Yourself) philosophy and the CRUD (Create Read Update Delete) system.
- **SQLite 3.9** is a really simple RDMS that is ACID-compliant. We chose SQLite due to the fact it's contained in a single file and is really light
- **HTML 5**
- **CSS 3**
- **jQuery** is a fast, small, and feature-rich JavaScript library.
- **Bootstrap 3** is the CSS framework used to design the web pages. It lets us have a sleek design with minimal efforts.
- **LinkedMDB** is the RDF database we chose to use to get informations about movies.

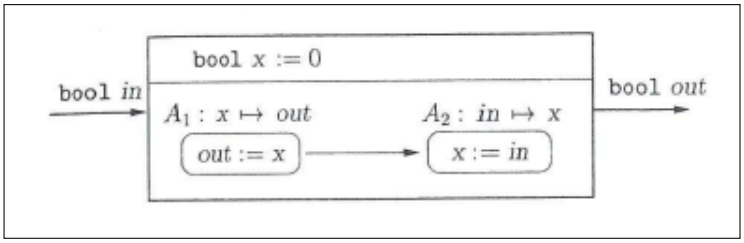

Figure 1: **Delay component**

3 Framework and MVC Pattern

3.1 Ruby on Rails

As stated earlier Ruby on Rails is a framework following the MVC pattern using Ruby. It is really simple to use and does a lot of work on the back end to let the developer fully focus on their code without worrying about every details like security or how to link different objects together.

In short, it gives a really high-level interface to developers to create a web application.

3.2 MVC

The Model-View-Controller (MVC) separates the view from the applications's logic.

3.2.1 Model

The model does the application's business logic. It has to handle data (storing and retrieving). In Rails the Model is implemented through Active Record which is a class that wraps the database access in a Ruby object for ease of use.

Active Record also have great tools that lets you handle the validation and the relationships between tables directly in the model. These validation and foreign keys are only present in the model layer this way and you will not have a foreign key present in the database.

In our project, the model layer was a bit more complicated to implement as we were using an external database namely the RDF database exposed by LinkedMDB. We had to retrieve information directly from LinkedMDB using the gem RDF.rb and link the movie to our Active Record movie object so we could take advantage of different possibilites given by Active Record. This was especially useful for all the recommendation part.

3.2.2 View

The view should only serve to encapsulate the presentation logic. We shouldn't do any application logic (Controller) or information retrieval through the database (Model). In our project, the view only display the movie information given by the controller.

3.2.3 Controller

The controller is the link between the view and the model. It will retrieve information from the model, apply some transformations to it and pass it to the view.

4 RDF

RDF stands for Resource Description Framework and is a W3C specification. It is a model used to link together web resources. The format to present such data is often N-Triples, meaning the information is separated in three pieces :

- The subject
- The predicate
- The object

RDF also links resources in a way that is easy for a computer to interpret as well asresources for a human. For example you can use the 'foaf' notation to indicate that two persons are linked. This notation is also transitive.

In our project we make use of such a RDF graph to search for movies using different criterias like the year or the actor.

5 SocialMovies Application

In this chapter we will look at different parts of our project that think are interesting.

5.1 Database and Models

The database in our project is mixed. We use models and SQLite to store the different lists, and to store the RDF movie ids as well as to store users. To get all the actual movie informations we query the LinkedMDB using SPARQL queries.

We have to store the movie ids in a model, because we need to somehow retain which movie a user liked so that can give him recommendations.

5.2 User Authentication

The user authentication is a critical part of a web application. It is essential to secure this part to the best of your capabilities.

To ensure that, we used a gem called Devise that manages authentication for us. This gem takes care of registration and signing in. It also stores the passwords in the database using strong encryption.

5.3 Movies

5.4 Recommendations

We wanted to be able to provide movies recommendations to a user based on what they liked. This can get pretty complicated as you have to create graphs and use data-mining techniques to emerge a relation graph between user.

5.4.1 Recommendable

Fortunately a gem called 'Recommendable' exists. It takes care of all the relations and just give you which item a user might like based on what they have liked previously.

In SocialMovies, you can either like, dislike or hide a movie. Based on these three actions, recommendable will generate a list of the recommended movies for you to see.

We had to use Redis and Sidekiq. Redis is a data-structure server working in-memory that is extremly efficient. Sidekiq is a gem that makes queues for background processes automated.

Redis was used to store the relationships between users for the recommendation part. Sidekiq was used to update the recommendations for each users. Obviously this task has to be done in the background otherwise the application would become really slow.

5.4.2 Lists

In our project we have implemented lists for a user to save movies they want to watch later, movies they have as favorites etc.

Users can add movies to the list they want and remove them at any time. The favorite list is also used to recommend movies.

6 Guides

6.1 Installation Guide

In order to run the web application you need to :

1. Install SQLite 3
2. Install Ruby ¿= 1.9.3
3. Install Rails 4.2.4
4. Install Redis
5. Install the gem Bundler
6. Copy the SocialMovies directory from the usb stick or clone it from https://github.com/jocelynthode/SocialMovies
7. Go to the SocialMovies directory and type :

   ```
   bundle install
   ```

8. Create the database by running in your terminal :

   ```
   rake db:create && rake db:migrate
   ```

9. start redis with :

   ```
   redis-server &
   ```

10. start sidekiq with :

    ```
    bundle exec sidekiq -q recommendable &
    ```

6.2 User Guide

To start using the app, you must register using the Sign Up button. After that you can start browsing and search for different movies. The more movies you like/dislike and add to your lists, the more precise the recommendations will get.

7 Conclusion

To conclude this report, we have implemented a web application that lets a user search for movies, get recommendations and use lists to save movies to watch later. We used Ruby on Rails + Bootstrap as the framework and RDF databases to retrieve movie informations.

We were already familiar with Ruby on Rails before this project, but had never used RDF. RDF has some really good ideas and is really generic which makes it easy to use in multiple completely different applications.

We still had problems with it, mainly due to the fact that most open link RDF databases are either out of date (LinkedMDB hasn't been updated for more than three years) or are missing informations. To palliate to these problems we had to rely on third party apis that aren't using RDF, such as OMDB.

Using SPARQL for the first time was also challenging, becuause it wasn't exactly like SQL which we are used to.

In the end we enjoyed working with Ruby on Rails and RDF and we wish there would be more services providing a SPARQL end-point and using RDF to organize their data.

References

[1] https://en.wikipedia.org/wiki/Resource_Description_Framework

[2] https://en.wikipedia.org/wiki/Ruby_on_Rails

[3] https://en.wikipedia.org/wiki/Model–view–controller