Percol8 - Coffee App

Matt Femmer
@femmermd on GitHub

Description

An application, created for coffee drinkers and baristas, that helps them to thoughtfully evaluate and track café beverages. Users can post photos, rate, and describe coffees in detail. Other users can discuss review posts in comments.

Features

- User profiles
- Coffee reviews with description
- Profile photos and review photos
- Landing page with feed of reviews
- Profile page with feed of user’s reviews
- User comments on reviews

Planning - User Stories

As a User, I can create a profile where I can post a short bio, profile picture, and see the coffee reviews I've posted to keep track of what I'm liking lately.

As a User, I can review the coffee I'm drinking, including a photo, tasting notes, descriptions of origin, brew process, and an optional story.

As a User, I can take part in the coffee community and discuss other users' reviews through commenting on their review posts.

Planning - Database

User Table

Id – id, generated value

ReviewList – ArrayList (one to many to Review table) of user's reviews

CommentList – ArrayList (one to many to Comment table) of user's comments

Username – String, unique

Bio – String, description chosen by user

Email – String

PhotoId – String, url generated by Cloudinary

ImageScale – int, used to set the size of image display

Planning - Database

Review table

Id – Id, generated value

User – User object (one-to-many to User table)

CommentList – Set of Comment objects (one-to-many to Comment table)

PhotoId – String, url generated by Cloudinary

ImageScale – int, used to set the size of image display

Review content String fields:

CoffeeName, DrinkType, Origin, Rating, BrewDevice, Roaster, Café, Flavors, Roast

Planning - Database

Comment table

Id: Id, generated value

Text: String, contents of comment

userId: int (many-to-one)

reviewId: int (many-to-one)

Technology Stack

- Java, HTML
- Spring framework
- Thymeleaf Template Engine
- MAMP database engine
- Cloudinary image management

Demo

What I Learned

- New controller structure – abstract controller with DAOs and general methods extended to Account and Review controllers
- User registration and password authentication
- Java session management (login / user in session / logout) and access to pages based on session state

What I Learned

- Accepting Image files through Thymeleaf forms, Image hosting on external server (Cloudinary) during post request handling, then retrieving image id from Cloudinary server to assign to review/user objects.

- Storing files directly in database by streaming from file to byte[ ] – feature later removed (not ideal for image hosting)

- Nested thymeleaf iteration for review feed and displaying review comments

What's Next

- UI improvements
- Likes
- Salt/Hash passwords
- Deploying website
- Mobile port