Mini project

Martin Pola

Purpose and goal

With the mini project, you should hopefully be able to apply many of the topics we have covered during the past two weeks in a more or less complete product. Either choose one of the suggestions below or talk to your Java trainer if you have idea for a project on your own.

The mini project is ideally carried out in groups of 2-4 people. It is strongly recommended not to work on a project alone.

The suggestions below have several iterations. That means you should be able to first get a *inimal viable product*, which you can then extend with other features. Between iterations, make sure to take a backup of your code, so you can go back to a working state should you have issues in a future iteration.

Project suggestions

The blog platform

Create a page where users can read and save posts to a blog platform. Each blog post entry could have a title and a body and blog post entries could be saved in files; one file per entry. It's wise to have a dedicated folder that is used only for blog post entries. Have one JSP file for reading the blog (listing all entries), one JSP file with a form for creating entries and a third JSP file to save entries that users submit through the form.

Date and time

Add timestamps to your blog post entries, in the files, and display them on the page where you can read blog posts.

Password protection

Don't let anyone create blog posts; protect the form with a password and require users to enter the password in the form when creating blog posts entries.

Additionally, add an *author* to each blog post, and make it so that the author name depends on which password the user entered. An idea could be to store a mapping of password and authors as a `Map` (`HashMap`) in the JSP file that handles submission of blog post entries.

Read blog posts on a separate page

Instead of displaying all blog post entries on the same page, only display the titles and let users click on a title to read the blog post. In a new JSP file, take the name of the blog post file a user wishes to read as part of the *query string* and, on that page, display only the blog post from that file.

The booking system

Allow users to book the washhouse (sv. *tvättstuga*) online! Have a form where users can choose a day using a `select` tag in HTML, and optionally add a comment. Let the user select any day between today's date and the next thirty days. Save the booking in a text file with the date as the name of the text file.

On another page, display a list of dates, starting today and spanning over the next thirty days. Under each date, list any bookings that are saved.

Unbook

Allow users to remove bookings by clicking a link that leads to a JSP file. Send the date of the booking in the *query string* and remove the file from disk if it exists.

Multiple bookings on a single day

Let multiple people book the same day. First, get it working without times, and then, perhaps also add times (e.g. 18:00-21:00). An idea for data storage could be to store all data for a single day in a single file, and separate multiple bookings using a special character (e.g. a \t – the tab).

Tips and tricks

Cookies Regardless of which project you choose, can you find a way to utilize cookies to simplify the experience for the user? For the booking system, perhaps you can let the user have a cookie with their booked date, to highlight that booking when they view the page.

Class with good-to-use methods Some things are very commonly used, but still require multiple lines of code. Examples could be reading from a file or generating a random string. Implement those lines of code as static methods in a class that you can then call from anywhere in your program, to quickly perform those common tasks. CommonTasks.java is an example of that. (Compile it and place it in `webapps/ROOT/WEB-INF`!)