Steve Babcock
7/20/2017

Blacklist Website

View live at http://67.149.120.212:8080/Blacklist

Sign in with walter@gmail.com and 'password' to access the admin account, or just create your own.

This project aims to allow market research firms to share information with each other. By submitting a tab-delimited text file containing the names, email addresses, zip codes, and phone numbers of respondents, information can be better kept up to date. The information in this database is spoofed for example purposes, and a respondent file is provided in this directory - blacklistData.txt.

This website was created with Eclipse Neon 2.2 and the Code folder can be imported into your workspace if you would like to build it yourself. It uses Apache Tomcat, the Java Spring framework, and an SQLite database. I often use this on the backend for convenience, but only a few lines of code are needed to move to a more production capable database such as SQL Server, MySQL, or Oracle.

The .db file is provided for SQLite and is ready to go. Place it in the bin folder of your Tomcat installation. Also, create a folder named uploads in the bin directory. This is where uploaded respondent files are stored.

The code is setup so that each .jsp file has a Java controller supporting it and receiving the requests coming from that page. Each controller extends the ControllerTemplate class to avoid redundancy. At the beginning of each controllers' post method, a User object is created from both the session and request. First from the session and then the request allowing for information to be updated properly if sent recently. This User object is also used to prepare the response which again avoids great redundancy.

The chat aspect of the website uses a long polling technique similar to Google Chat. Raw tcp sockets are a work in progress for the browser environment. So this employs the familiar ajax calls on the client side made in succession. On the server, a new thread is spawned for each

request, and will sleep until a message has arrived for that user (max eight seconds). This creates the illusion of server-side initiation using reliable web standards.