Alex McLaurian

4-6-2014

CS 5560

Hackathon

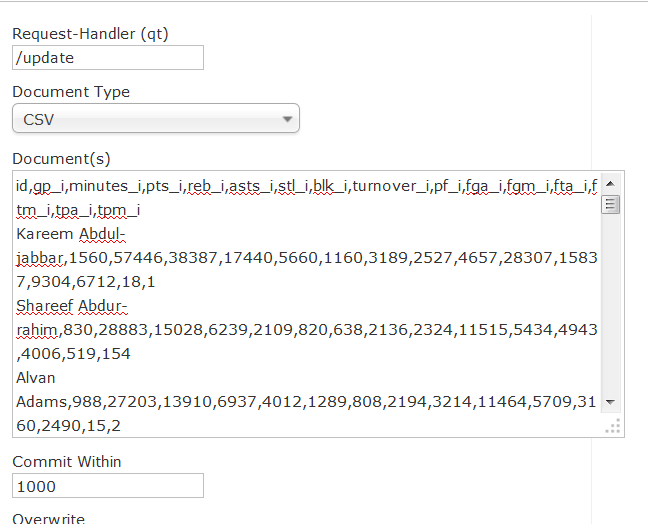
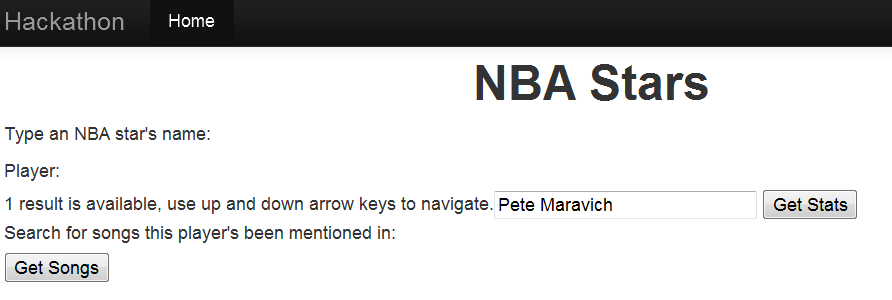
**1. Design**

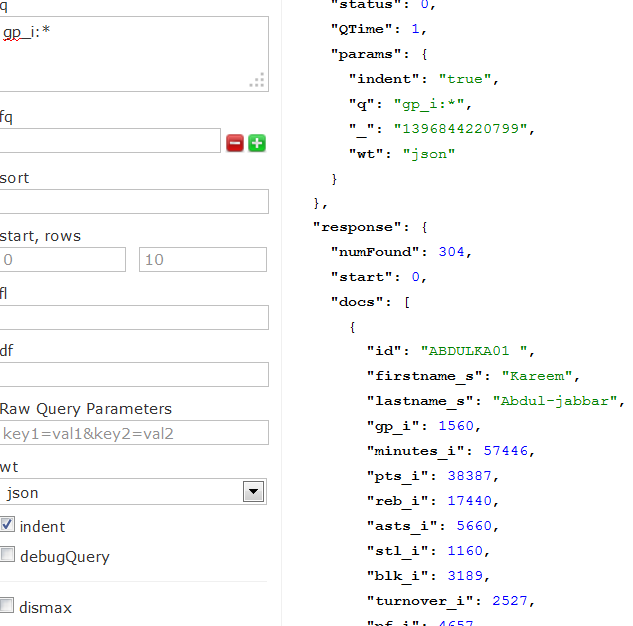
* My idea for the hackathon was to analyze the top NBA basketball players, and determine their cultural impact by seeing how many times they were mentioned in songs.
* I used a dataset from [www.databaseBasketball.com](http://www.databasebasketball.com) for my main basketball player data, which contains career statistics for all players, and musixmatch.com for song lyric lookups. The integration involved was to first receive the user’s desired basketball player to lookup, then give options for either statistic retrieval via Solr or looking to see if they are mentioned in any songs via MusixMatch.
* The basketball dataset contained over 3,000 players, but to ensure a higher hit rate in terms of finding players who would be mentioned in songs, I limited the player pool to over 300. The players were selected based on games played and points scored. I would have liked to integrate twitter mentions and endorsement deals along with current and older players into a more predictive algorithm to determine which rookie players could have greater cultural impact going forward, but I wasn’t able to reach this point.
* My mobile design is very simple, utilizing Twitter bootstrap. I incorporated an autocomplete function along with the search bar to ensure users search only for players I have added to Solr.

**2. Features Implemented**

* The Solr Index/MusixMatch API queries are both based on the same player name. The next step would be to retrieve the total song mentions via the API and insert into the appropriate Solr document, building up a more truly integrated, dynamic Solr index.
* I utilized the Solr index, uploading documents with my own unique schema. This was a roadblock at first as I didn’t know how to add new fields, but this issue was resolved. The ids are based off the player’s names, and the remaining fields all deal with specific statistical categories. They are named the following: id,gp\_i,minutes\_i,pts\_i,reb\_i,asts\_i,stl\_i,blk\_i,turnover\_i,pf\_i,fga\_i,fgm\_i,fta\_i,ftm\_i,tpa\_i,tpm\_i
* I utilized Eclipse and Glassfish for development, as well as Java Web Services
* I used Twitter Bootstrap, AJAX and JQuery for user interface development

**3.Outputs: description with screenshots of the Features**





**4.All the Web Service and Web SiteURLs**

**5.**[**https://github.com/aemz86/CS560/tree/master/Exam**](https://github.com/aemz86/CS560/tree/master/Exam)

**6.Limitations**

My project was limited by my confusion with implementing Mahout. While the tutorials provided great information, I was unsure of how to approach using my dataset in a classification or clustering algorithm. Writing the code to format the data and achieve logical results in Mahout was difficult. Additionally, while I could successfully call Solr and the MusixMatch API via REST services, displaying the retrieved JSON had challenges. I think this may have been due to my attempting to run the AJAX queries locally instead of on any external server.

**7.References**

Solr information that was helpful: [http://heliosearch.org](http://heliosearch.org/)