Vlad Predovic

CS 496

Due 10/26/2015 (one extra day granted on request)

HW3 Part 2

An explanation of the URL structure used to access resources

At this moment in time, the only possibility with the API is to access, add, and delete the existing information on the Datastore. However, these basic features would be the most important aspect of the app I am designing. The URL structure is very simple.

The API is capable of the displaying in bulk format the existing entries in the Datastore along with their details. To do this navigate to one of the following (depending on what you are searching for).

http://proj2-1095.appspot.com/studyProcedures

http://proj2-1095.appspot.com/TimeStudies

http://proj2-1095.appspot.com/TSinstance

Simple get procedures are in place to display all existing entries. The application API interface was built this way because to utilize the TimeStudy app correctly, a user could easily benefit from querying my different instances. Therefore it is beneficial to be able to scan through a list of entries, especially under TimeStudies, in order to access the associated TimeStudy instances and procedures.

A description of which RESTful constraints you met and which you did not (you do not need to make a RESTful app, but you do need to know why it does not meet all the constraints of a RESTful API)

Statelessness: In the body of any request made all the parameters and data are sent to the server needed to generate a response.

Client-server: Concerns of client and server are separated. The server is able to interface with multiple clients, while the client does not have any need to see the inner-workings of the server. Most of this is achieved through use of the google-app engine.

Stateless: No sessions, all input by the client has all information provided to do the necessary tasks. Changes in resources are saved, but cannot be tied to the client.

Layered System: Client has no need to understand the different layers of the system. All functions interact with server to complete client's task without his knowledge of the interaction.

Uniform Interface: The client can only request JSON representations at the moment and all representations on the API live site are in text/html format. Although caching was attempted (see timestudy.py) it does not have a use for my server use and the no-cache default setting is the most to my advantage as up to date data is very important. Future use would be setting a cache to final product images for my TimeStudy Ndb model.

All four of the HTTP requests are used. Gets are used to present bulk data on every page. Posts are used to add new tuples to the Datastore, also available for all three model types. PUT is primarily used to append data. Finally DELETE is used to wipe a TimeStudy and all associated instances and procedures.

A discussion of any changes you needed to make to the schema from last week

There were a few critical changes that needed to be made to my schema in order for it to function smoothly. These changes were primarily made so that each entity could also function autonomously. Coming from a background in relational databases, I can see why these were my primary pitfalls. The first change I made was that I added the final figure to the TimeStudy entity and removed it from the TSInstance entity. This was done because it makes no sense to upload the same image multiple times for separate instances of the same study. That is just a waste of space.

Another critical change was adding study procedures to the TimeStudy entity. The logic behind this issue was that the Procedures should be directly tied to the study. I added rigidity to my database so that you would not be able to change procedures for each TimeStudy based on the instance. The only candidate keys in a Time Study Instance are the person conducting it and the date that the study was conducted on.

Although my model has a lot of issues, there is a key aspect that still allows it to function properly. This is the attribute that was added to the instances and procedures called 'DateTimeStarted'. Every time a new instance of the procedures is created, this attribute grabs the current time stamp which is exact to the second. This is important because it allows one to group procedures that were ran under the same instance because they will have the same iteration.

Having finished the API, what you would have done differently

I spent a large amount of time restructuring the database and was not able to devote as much as I wanted to on the API. Given the opportunity to do It again, I would have done a better job of connecting the different entities together. Most importantly, restricting the ability to make TSinstances unless the TimeStudy exists, and immediately creating new procedures with the existing TSinstance's instance and time stamps. A lot of these nuances would have made the interactions with the API a lot smoother.