Group Members :: Atkins, Brandon;

Carter, Adam;

Kim, Yong; and

Swann, Matthew J

Re :: Comp 6000: Web Development – Formal Design

Date :: 2013-03-25

Project Title :: Interactive Database Backup, Import and Migration Tool utilizing

AJAX and XML to support web-hosted databases written in the Django

Framework

As mentioned in the proposal for this project, the goal of the experiment is to develop an interactive database tool for data import, export and migration using XML as a medium. The implementation will require string parsing and conversion algorithms. These are currently designed to work similarly to other data serialization mechanisms.

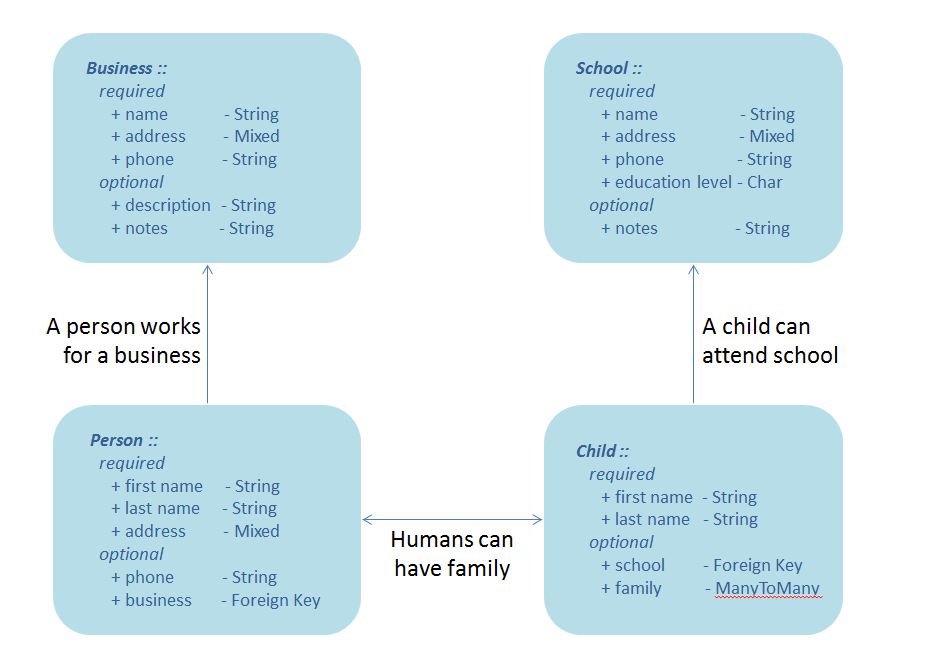
AJAX returns will pull various pieces of data across multiple tables from within the website. The user will then have the ability to re-query the database or output the resultant query-set to XML with a time-stamped file name to allow for the existence of a plurality of documents and also for document tracking and data validation. This will allow for a more conclusive diagnosis as to the effectiveness of this mechanism in comparison to other like implementations.

Interaction with the database will include queries normalized in function, but fluid enough such that string search data can be referenced against multiple DB tables. Import and output functionality will take place via custom string parsing mechanisms that prevent duplicative data insertions or storages.

The database design will follow standard MVC structure. Database tables and schemas will be modeled as will data behavior and user interaction. This abstraction will represent one of the project’s three major code packages. Another package will be comprised of the physical data itself. This data will exist in numerous forms including but not limited to an XML format, a JSON format, and Django’s natural SQL storage. This package will also include the custom functionality for XML import, export and migration. The final code package will include the files associated with database administration and configuration.

Data validation will be necessary. As such, formalized unit-testing frameworks will be established in order to compare the imported and exported results to hardcoded data entry. Data validation will prove the viability of the concept. A comparison to the computational time associated with other mediums of data transactions will answer the question as to whether the hypothesis is indeed a reasonable practice.

PHYSICAL DATA\_BASE DESIGN :: ( Figure 1 )



The database will be comprised of four primary tables: Business, School, Person and Child. A fifth meta table will exist to express the relationships between children and people; however, Django’s natural meta-table behaviors will be used to create same.

STRUCTURAL PROJECT DESIGN :: ( Figure 2 )

|  |  |
| --- | --- |
| folder.JPG #Documents | administrative file for project related documents |
| Package.JPG Comp6000 | database configuration package |
| Python.JPG settings.py | database settings, configuration and organization file |
| Python.JPG urls.py | url mapping |
| Python.JPG wsgi.py | wsgi script |
| Package.JPG Control | physical data interaction package |
| folder.JPG fixtures | physical data representations… |
| folder.JPG json | … in json |
| Text.JPG database\_testdata.json |  |
| folder.JPG xml | … in xml |
| Text.JPG xml\_import\_file.txt |  |
| Text.JPG xml\_export\_file.txt |  |
| folder.JPG templates | html templating |
| Text.JPG *{un-named html file}* | *~ one file per page* |
| … … … |  |
| Text.JPG *{un-named html file}* |  |
| Python.JPG admin.py | script for admin access and direct data interaction |
| Python.JPG xml\_functions.py | scripts for xml importing and exporting |
| Package.JPG Data\_Base | database behavior and modeling package |
| Package.JPG Data | hardcoded data import package |
| Package.JPG Imports | individual import package and scripts for … |
| Python.JPG business\_import.py | … the business table |
| Python.JPG child\_import.py | … the child table |
| Python.JPG person\_import.py | … the person table |
| Python.JPG school\_import.py | … the school table |
| Python.JPG Prime.py | primary import script |
| Python.JPG models.py | database table schemas and functionality |
| Python.JPG tests.py | data verification via formalized unit-testing |
| Python.JPG views.py | database interaction medium; bridges the html to the  database tables |
| Python.JPG manage.py | primary database control script |
| Text.JPG sqlite.db | django’s inherent physical database representation |