Resources / Assignments (/COMP9321/18s1/resources/15672)
/ Week 6 (/COMP9321/18s1/resources/15674)
/ Assignment 2: Implementing a Data Ingestion and Publication Service

Assignment 2: Implementing a Data Ingestion and Publication Service

- Deadline: Week 8 Thursday (26th April) 23:59.
- Total 10 marks allocated.
- Individual assignment.
- Late penalty: 20 percent of the final mark per day

Specification | Make Submission | Check Submission | Collect Submission

Data Service for NSW Crime Statistics

In this assignment, you are asked to develop a data service that allows a client to read and store some publicly available crime statistics data, and publish the content in a standard Web data publication format.

The Data Sources

IMPORTANT: For this assingment, you will be asked to access the given Web content programmatically . Some Web hosts do not allow their web pages to be accessed programmatically, some hosts may block your IP if you access their pages too many times. During the implementation, download a few test pages and access the content locally, i.e., try not to perform too many HTTP GET programmatically with the site.

Data Source 1 - NSW Bureau of Crime Statistics and Research

The source URL: http://www.bocsar.nsw.gov.au/Pages/bocsar_crime_stats/bocsar_lgaexceltables.aspx (http://www.bocsar.nsw.gov.au/Pages/bocsar_crime_stats/bocsar_lgaexceltables.aspx)

The individual LGA links (e.g., Blue Mountains): http://www.bocsar.nsw.gov.au/Documents/RCS-Annual/bluemountainslga.xlsx (http://www.bocsar.nsw.gov.au/Documents/RCS-Annual/bluemountainslga.xlsx)

This page publishes NSW recorded crime statistics organised by the names of Local Government Areas (LGA). Each link is an EXCEL file. We are only interested in the content of the first sheet "Summary of Offences". You can also ignore the footnotes at the bottom of the first sheet (i.e., after "Other offences")

The URLs for individual LGAs have a distinctive pattern. You can use the pattern to construct a URL for a particlar LGA.

The content of the file is quite straightforward. If you want to understand the exact terms and definitions, refer to the definitions and explanations page. (http://www.bocsar.nsw.gov.au/Pages/bocsar_crime_stats/bocsar_glossary.aspx)

The data service specification will ask you to 'import' this data into a 'data storage'. You should inspect the content of the file carefully and decide on a data model and storage method. You should represent everything in the first sheet of the excel file (except the footnotes).

You will find that either a JSON or XML model (i.e., document-based model) is suitable. You could use either MongoDB or XML files as your storage.

Data Source 2 - Australian LGA postcode mappings

The source URL:
https://docs.google.com/spreadsheets/d/1tHCxouhyM4edDvF60VG7nzs5QxID3ADwr3DGJh71qFg/edit#gid=900781287
(https://docs.google.com/spreadsheets/d/1tHCxouhyM4edDvF60VG7nzs5QxID3ADwr3DGJh71qFg/edit#gid=900781287)

Originated from: http://greenash.net.au/thoughts/2014/07/australian-lga-to-postcode-mappings-with-postgis-and-intersects/ (http://greenash.net.au/thoughts/2014/07/australian-lga-to-postcode-mappings-with-postgis-and-intersects/)

This Google Doc page contains the mapping of Australian postcodes to their respective Local Government Area (names). This mapping data allows you to figure out to which LGA a postcode belongs. You only need to consider the NSW postcodes.

This data source is used by the data service internally as a look-up/reference information. Since this is not directly accessed by the client, it is up to you to decide how you store and access this data for your data service implementation. You'd probably need to read and process this once.

Data Service Specifications

The data service will use ATOM and JSON format to publish its data. To generate a correct ATOM format, you should read some documentations on ATOM yourself. A page like this helps (https://validator.w3.org/feed/docs/atom.html) and contains enough for you to design a correct format.

In the specification below, ATOM is shown as output. You should also support JSON equivalent in all outputs. The client will use *HTTP Content Negotiation* to choose the format it needs.

The service implements the following operations:

Creating a data entry with the data service

This operation can be considered as an on-demand 'import' operation. The service will download the excel file identified by the LGA name given, process the content of the first sheet into an internal data format and store it in the internal database (MongoDB or XML files)

The interface looks like as follows:

- HTTP operation: POST /[your_collection_name]
- Input Payload (as form parameters):

```
lgaName="Blue Mountains"
or
postcode="2745"
```

- Returns: 201 Created

```
<entry>
   <id>[your_collection_address]/{id}</id>
   <title /> [appropriate title]
   <updated>2018-04-08T12:06:11Z</updated>
   <author>
      <name />
   </author>
</entry>
```

The return response contains the location of the data entry created as the result of the processing the import. What is shown above is just a suggestion. You can decide what else is appropriate. You do not have to return the actual imported content with POST. Just return the location (ID).

If an input contains a postcode instead of an LGA name, you should use the LGA-postcode mapping data (Data Source 2) to identify the matching LGA name. Note that the given mapping is not up-to-date. If the input postcode maps to an LGA name that doesn't exist in Data Source 1 return error 400.

If an input contains an LGA or postcode that already has been imported before, you should still return the location of the data entry - but with status code 200 OK (instead of 201 Created).

A POINT TO PONDER ABOUT: An `asynchronous POST'?? If a POST takes too long, you may not want the client to wait. What would you do to solve the problem in a RESTful manner?

Deleting a data entry with the data service

This operation deletes a data entry

The interface looks like as follows:

- HTTP operation: DELETE /[path_to_id]
- Returns: 200 OK

Retreiving the available collection with the data service

This operation retreives all available data entries (i.e., the collection). The response of this operation will show a list of available LGA data entries, some of their metadata and links (IDs) of each data entry. This is very similar to what AtomPub Service Document does.

The interface looks like as follows:

- HTTP operation: GET /[path_to_the_collection]
- Returns: 200 OK

```
[ ... list of ATOM entries and metadata ...]
```

Retreiving a data entry with the data service

This operation retreives a crime statics data entry. The response of this operation will show the imported content of the excel file. That is, the data model that you have designed is visible here inside an ATOM entry's content part.

The interface looks like as follows:

- HTTP operation: GET /[path_to_id]
- Returns: 200 OK

```
<entry>
   <id>[your_collection_address]/{id}</id>
   <title />
   <updated>2018-04-08T12:06:11Z</updated>
   <author>
      <name />
   </author>
   <content type="application/xml">
      [your data model for the LGA crime statistics goes here]
   </content>
</entry>
```

Retreiving data entries with filter

Implement REST-based Data Access Service serving the following queries (roughly inspired by OData syntax).

You do not need to implement all possible scenarios for data query operations. In fact, we ask you to implement only two types of queries. The syntax (in BNF form) is explained below.

```
<query-type-one>  ::=    <lga-filter> | <lga-filter> <or-operator> <lga-filter>
<lga-filter>      ::=    <lga-name> <eq-operator> <lga-value>
<or-operator>     ::=    or
<eq-operator>     ::=    eq
<lga-name>        ::=     ? alphabetic characters and space ?
<lga-value>       ::=     ? alphabetic characters and space ?
<query-type-two>  ::=    <lga-filter> <and-operator> <year-filter>
<and-operator>    ::=    and
<year-filter>     ::=    <year-name> <eq-operator> <year-value>
<year-value>      ::=     ? 4 digits year value?
<year-name>       ::=     ? alphabetic characters and space ?
```

The above query syntax is added after a URL query parameter named `filter'. For example, the following shows possible use of the query syntax:

- URL: http:// .../[data_collection_root]/filter?lgaName eq Bayside or lgaName eq Ballina
    - Returns both the Bayside and Ballina crime statistics (two entries)
- URL: http:// .../[data_collection_root]/filter?lgaName eq Bayside and year eq 2014
    - Returns the Bayside crime statistics including only 2014 column.

Authentication and Authorisation

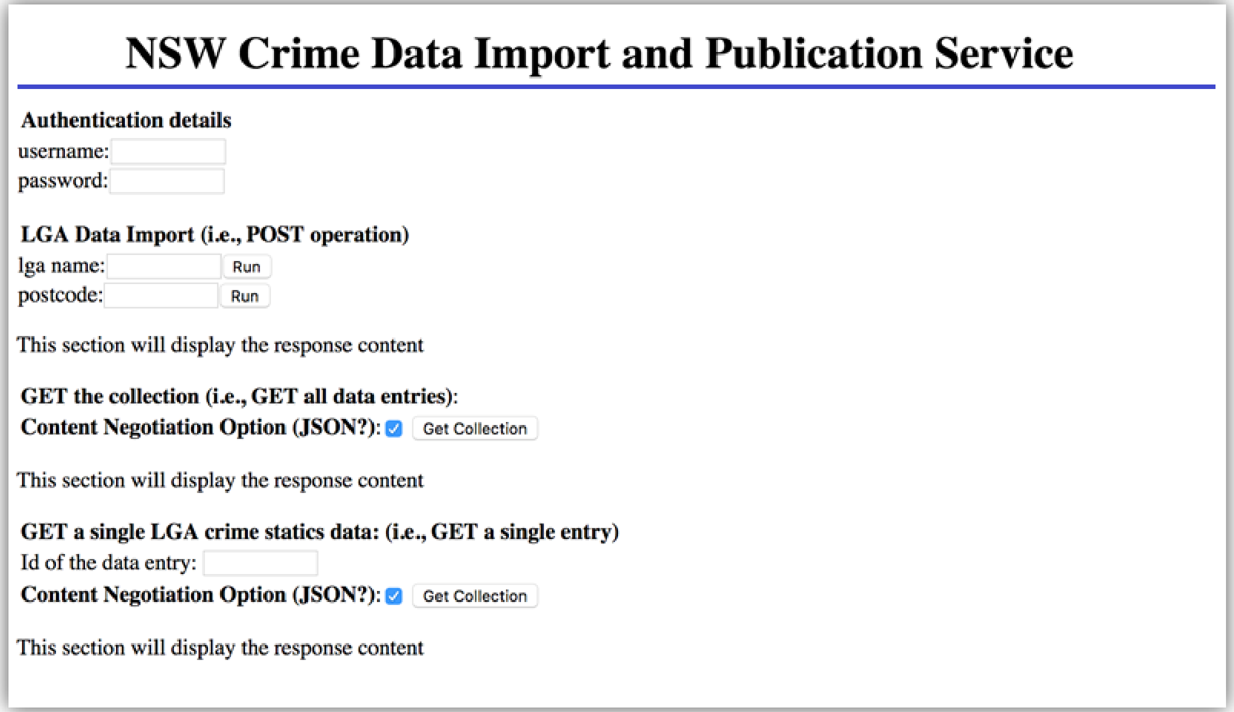
Your data service must implement authentication and authorsation for access. There are two types of users: admin, guest. Their passwords are admin, guest respectively.

- For POST and DELETE, the client should be authenticated and authorised to be admin.
- For all GET operations, the client shoudl be authenticated and authorsed as either admin or guest.
- For implementing this, refer to the lab exercise on token-based method for authentication and authorisation.

A Simple Testing Client

Either as a *Python Flask Web application* , or as a *JavaScript-based Single-Page Application (See Lab 04 (https://webcms3.cse.unsw.edu.au/COMP9321/18s1/resources/15817))* , you should provide an implementation for a client app for testing. Please be noted that if you want to create a Flask Web Application, you must connect to the RESTful APIs; in other words, the client will not contain any business logic, it must only make API calls to the server.

The implementation should make **the testing of all operations possible and give at least an example to show how to test your RESTful APIs.** The screen below is included just to give you an idea ... but it is only a suggestion. You can design your own that is suitable for the purpose.


NSW Crime Data Import and Publication Service

Authentication details
username:
password:

LGA Data Import (i.e., POST operation)
lga name: Run
postcode: Run

This section will display the response content

GET the collection (i.e., GET all data entries):
Content Negotiation Option (JSON?): ✓ Get Collection

This section will display the response content

GET a single LGA crime statics data: (i.e., GET a single entry)
Id of the data entry:
Content Negotiation Option (JSON?): ✓ Get Collection

This section will display the response content

Submission Instructions

The marking environment setup is basically the same as the lab setup. We will use PyCharm to import and run your solution. You must make sure that it runs correctly that way before making a submission. We will not be tweaking any of the configuration.

Prepare the following files for submission:

1. You need to make a zip file of your server (the main project which does the business logic) `server.zip`
2. You need to make a zip file of your client (either a web application or a JavaScript client) `client.zip.`
3. README.txt (installation guide) :

```
// README.txt
Student ID: Name:
Installation Guide
- How to compile, deploy the service code (step-by-step)
- Specify if there is anything else you want to mention to help the assessor
```

Submit the prepared files (server.zip, client.zip, README.txt) through 'Make Submission' tab at the top of the page.There are couple of notes that you should pay attention before starting your project.

- If you use XML files to store data, then make sure that you are not using **absolute paths** . if you are using MongoDB, you are not allowed to use a local DB, follow the instructions in Lab 07 to see how to use Database-as-a-Service via mLab (https://mlab.com/) .
- For this assignment, you do not need to submit your virtual environment; instead follow the instructions in Lab-06 (https://webcms3.cse.unsw.edu.au/COMP9321/18s1/resources/16047) and use **requirements.txt** . Tutors will install all the requirements using this file.

Resource created 11 days ago (Tuesday 10 April 2018, 08:57:05 AM), last modified 9 days ago (Thursday 12 April 2018, 09:15:17 PM).