CO3098/CO7098
Coursework 3

Mini Web Project

Important Dates:

Handed out: 25-Nov-2017
Deadline: 22-Dec-2017 at 23:59 GMT

The deadline is strict and will not be changed. Please ensure that you submit your work in time.

- This coursework counts as 22% of your final mark.
- Please read guidelines on plagiarism in the study guide and course documentation.
- This coursework requires knowledge about Bootstrap, jQuery/AJAX and the Web MVC framework

Coursework Description

MyBookmark Explorer (MBExplorer) is an online bookmark manager that can be used to store your favourite items. There are three types of bookmark items that can be added: (1) ***links*** (2) ***text files***, and (3) ***locations***. A user can create folders and sub-folders and uses them to organize items. Your task is to design and implement a web interface and necessary RESTful web services backend for MBExplorer.

Note: some Java domain class templates are provided (see Appendix 1.3). You may use these classes for the implementation, and you are allowed to change them. However, you may choose not to use them.

Part 1: REST API [35 marks]:

MBExplorer allows users to create and edit bookmark folders, as well as organise them into a hierarchy. Your task in part 1 is to implement the following REST API to provide backend support for the web interface.

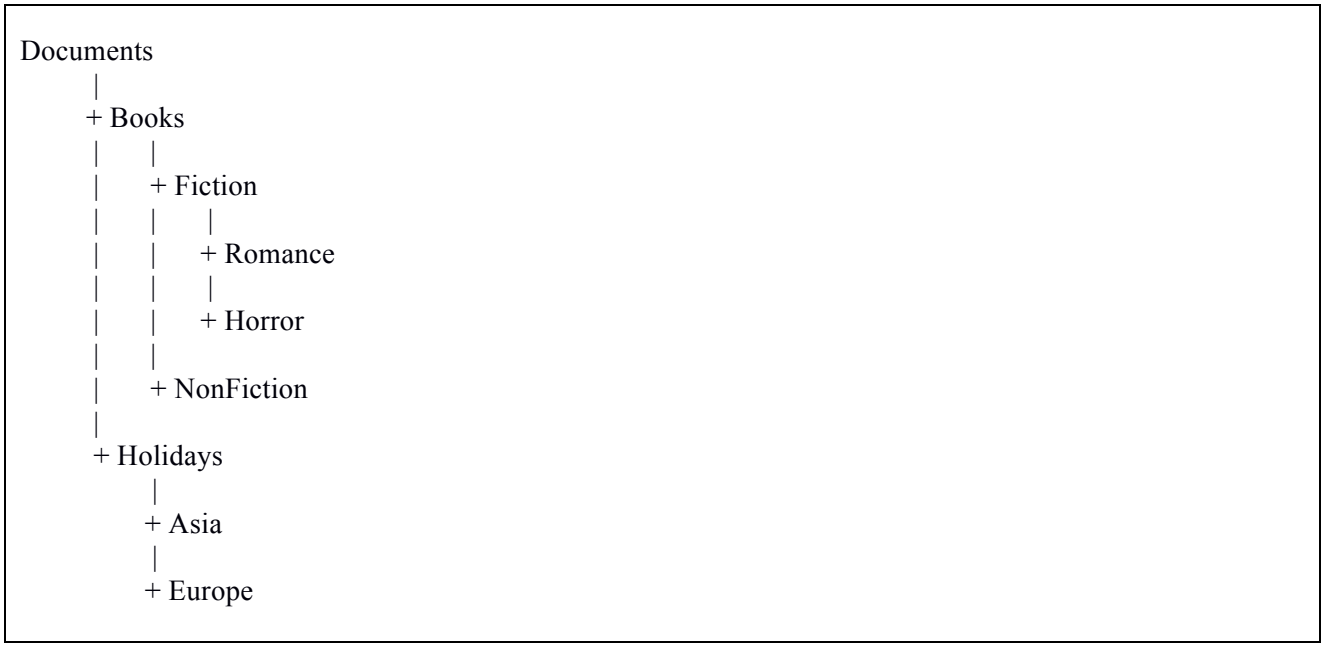
(a) `GET /service/create?folder=Romance&parent=Documents|Books|Fiction`

Add a new bookmark folder to an existing folder. This request returns `true` if the operation is successful, otherwise `false` are returned (if a folder with the given name already exists or the parent folder does not exist). If the parent folder is not explicitly specified then the folder to be added will become a top-level folder. Consider the example above. This request will create a new folder called `Romance` inside `/Documents/Books/Fiction` folder.

Consider the following HTTP GET requests:

```
GET /service/create?folder=Documents
GET /service/create?folder=Books&parent=Documents
GET /service/create?folder=Holidays&parent=Documents
GET /service/create?folder=Fiction&parent=Documents|Books
GET /service/create?folder=NonFiction&parent=Documents|Books
GET /service/create?folder=Romance&parent=Documents|Books|Fiction
GET /service/create?folder=Horror&parent=Documents|Books|Fiction
GET /service/create?folder=Asia&parent=Documents|Holidays
GET /service/create?folder=Europe&parent=Documents|Holidays
```

The following folder structure will be created



(b) `GET /service/delete?folder=Documents|Books`

Delete a bookmark folder from the directory structure. Any sub-folders or items kept in this folder (including items in its sub-folders) will be deleted too. For example, the request above will delete the folder "`Books`" and all its sub-folders including "`Fiction`", "`Romance`", "`Horror`" and "`NonFiction`", and any items inside them will also be deleted. The request should return `true` if the operation is successful, otherwise `false` is returned.

(c) `GET /service/structure?folder=Documents|Books`

Return a hierarchical JSON string reflects the bookmark folder structure. For example, the request above should return a JSON object in response:

```
{
       "folder": "Books",
       "subfolders": [{
       "folder": "Fiction",
       "subfolders": [{
              "folder": "Romance"
       }, {
              "folder": "Horror"
       }]
       }, {
       "folder": "NonFiction"
       }]
}
```

`(d) GET /service/count?folder=Documents|Books`

Count the number of direct and indirect sub-folders inside a given folder.

```
{"direct":"2","indirect":"4"}
```

`(e) GET /services/createStructure?tree=[Books[Fiction[Romance|Horror]|NonFiction]]&root=Documents`

This service constructs a sub-folder from the given string. For example, the above GET request adds a sub-tree hierarchy to the "`Documents`" folder. The service returns `true` if the operation is successful, otherwise `false` is returned (Hints: write a simple string parser using recursion or stack)

Note: You will need to implement other service to complete Part 2 and Part 3.

Part 2: Web Interface [50 marks]:

The web interface should support the following features:

Folders

a) Display the folder structure in a TreeView. (See **Appendix I** for more information on TreeView)
b) Create, edit and delete folders from the TreeView.
c) Show selected folder details (total number of items inside the given folder, including those in all its sub-folders, permission)
d) TreeView should be refreshed when the folder structure changes.

Items

e) Create, view, edit and delete bookmark items inside a folder from the TreeView. Apart from the operations with folders, the user should be able to create bookmark items. There are three kinds of items: (1) Links, (2) Text files, and (3) Locations.
    1. *Link* - (*title* and *URL*)
    2. *Text file* - (*title* and plain text *content*)
    3. *Location* (*Place name* and its *GPS coordinate*) - Locations should be displayed on Google Map

    (See **Appendix I** for the Java domain classes and Google Map API)

Miscellaneous

f) Lock/unlock folder – recursively set/unset read-only attribute for all items and its sub-folders.
    1. No item can be added to the folder if it is set to ready-only.
    2. An item becomes not editable when it is read-only.
g) Search folders/items, filter by item types and keywords (title and content)
h) Move folders/items.

Assume that

- Each folder and bookmark item must have a unique name (items or sub-folder in different folders may have the same name).
- Deleting a folder will also delete all items within the folder (and items in its sub-folders).

Part 3: Visualisation [15 marks]:

Use Google Charts API to visualise the folder structure in **Treemap** and **Organisation chart**.

a) Treemap: The folders should be displayed as rectangles, where the sizes of the rectangles represent the number of items inside this folder. See the link below for more information.

b) Ogranisation chart: Display the hierarchy of the folder structure (folder only)

(See **Appendix II** for Java domain classes and Google Chart API)

Bonus Question [10 marks]

Syncs the bookmark folder structures and items to a Dropbox account, where

- Links should be exported as *.url*
- Text file items should be saved as **.txt*
- Locations should be saved as *KML*

(See **Appendix III** for more information on Dropbox API and KML format)

Submission

- Zip all files in a single zip file for submission:
    - Your Dynamic Web project or Gradle/Maven project folder
    - README.txt
    - Your SQL schema and data, if applicable (Your_email_ID.sql)
- The archive should be named **CO3098_MiniWeb_email_id.zip** or **CO7098_MiniWeb_email_id.zip** (e.g. CO3098_MiniWeb_yh37.zip).

Note: Please contact the module convenor first if you choose to use .NET WCF or other frameworks for Part 1 and 2.

Your submission should also include a completed coursework plagiarism coversheet and a self-evaluation form (print and signed PDF or image). You need to submit the zip file via Blackboard and you are allowed to re-submit as many times as you like **before** the deadline. Marks for any coursework, which does not have the accompanying cover sheet, will be withheld till you provide one.

Mark Scheme

Part 1 [35 marks]
Part 2 [50 marks]
Part 3 [15 marks]
Bonus [10 marks]

Note: Bootstrap or any other web front-end frameworks may be used for Web UI design. jQuery AJAX should be used when necessary. The responsiveness and the usability of the web pages will be taken into account when grading Part 2 and Part 3. You may use any Web MVC Frameworks for implementation, including but not limited to:

- Spring MVC
- ASP.NET MVC
- Ruby On Rails
- Laravel
- AngularJS
- Django

Please consult with the convenor if the web framework you intend to use is not listed here. The system architecture and good coding practice will also be taken into account when allocating marks. The mark for this coursework will be capped at 65% if the solution does not use any Web MVC framework.

There is a bonus questions, you can earn extra bonus marks, but the maximum mark for this assignment is 100%. Extra bonus mark above 100% will not contribute towards to the final grade.

Appendix I

1.1. You can use any CSS/JS library for showing the TreeView. Here are some possible candidates:

Bootstrap TreeView:
http://jonmiles.github.io/bootstrap-treeview/
jqTree
http://mbraak.github.io/jqTree/#demo
JSTree
https://www.jstree.com/demo/
jQuery SONSAI
http://simonwade.me/jquery-bonsai

1.2. Persistence Frameworks

You may use **MBE.sql** provided on the blackboard, if you intend to write your own DAO classes and use some ORM frameworks such as Hibernate for data persistence. **You do NOT have to use this file** if you use Spring JPA, NoSQL (e.g. Mongodb, Oracle Spatial Graph) or other persistence frameworks.

1.3 Domain classes

You may use the Java domain classes attached for the implementation. However, **you may choose NOT to use them**.

```
BookmarkEntity.java
Folder.java
Item.java
ItemLink.java
ItemLocation.java
ItemTextFile.java
```

Appendix II

Google Map API

API Key:
https://developers.google.com/maps/documentation/javascript/get-api-key
Simple place marker:
https://developers.google.com/maps/documentation/javascript/examples/marker-simple

Google Chart API

TreeMap:
https://developers.google.com/chart/interactive/docs/gallery/treemap
Organisation Map:
https://developers.google.com/chart/interactive/docs/gallery/orgchart

Appendix III

Dropbox API: https://www.dropbox.com/developers/documentation
KML format: https://developers.google.com/kml/documentation/