

## Assignment 3

### Web Application Development

**Due date: 9:00pm Friday 30 May 2014 -- Worth 10%**

[Late submission will be penalized, 10% per working day for at most 5 working days]

#### 1 Assignment Description

This is an individual assignment. Students are referred to the Faculty's policy on plagiarism. The aim of this assignment is to develop a better understanding of building web applications from existing Web Services / APIs using Asynchronous JavaScript and XML techniques.

#### 2 Assignment Tasks

The main task of this assignment is to implement several functions for a simple accommodation search system. You are required to use the core Ajax technologies (JavaScript/HTML, DOM, XML, XMLHttpRequest, CSS, and XPath/XSL) and PHP to implement several functions of such a system.

The rental property data can be obtained from a real estate agent website and is supposed to be organized in an XML file. The data in the data file will be used with the Google Maps API to create a map with markers that contain accommodation information for rental properties that satisfy certain search criteria.

The various tasks that must be completed for this assignment are specified below.

##### 2.1 Rental property data collection and XML transform

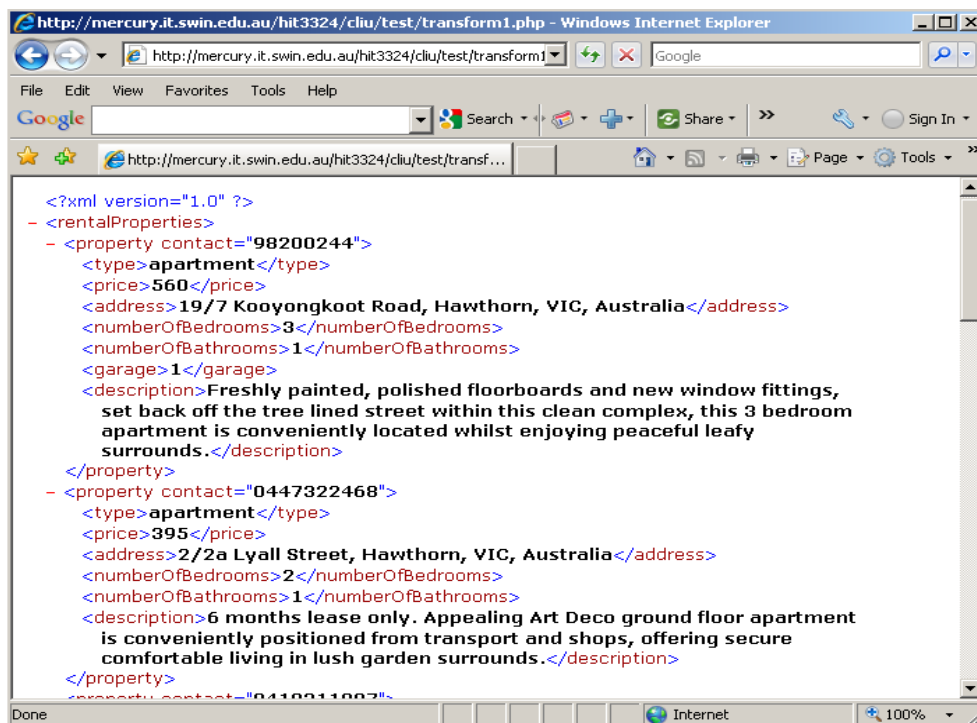
You are required to manually collect the data from a real estate web site such as *www.myhome.com.au* or *www.realestate.com.au*. The rental property data is to be stored in an XML document *rental.xml* with the following structure:

```
<rentalProperties>
  <property available="yes or no" contact="phone number of the agent or the owner">
    <type>house, apartment, unit, townhouse, etc. </type>
    <price>price per week</price>
    <address>
      <streetNo>111 or 2/222</streetNo>
      <street>say, Burwood Road</street>
      <suburb>say, Hawthorn</suburb>
      <state>VIC as the default</state>
      <zipcode>say, 3122</zipcode> /* optional
    </address>
    <numberOfBedrooms>1, 2, ...</numberOfBedrooms>
    <numberOfBathrooms>1, 2, ...</numberOfBathrooms> /* optional
    <garage>1, ...</garage> /* optional
    <description>... ..</description>
  </property>
  .....
</rentalProperties>
```

Your tasks are:

- 1) Collect data for at least 20 rental properties in Victoria and represent them in the above XML document. Make sure that the collected properties include at least 15 available apartments or units. These apartments/units must all have 2 or more bedrooms and be located in at least 3 different suburbs. Once you have the XML document, put it on the mercury server (in *www/htdocs* instead of *www/data*) for later access via the url <https://mercury.ict.swin.edu.au/cos80021/<username>/Assignment3/rental.xml>.
- 2) Perform a server-side transformation (*transform1.php*) on the *rental.xml* using an XSLT style sheet called *apartment.xsl*. The result of the transformation should be an XML file called *apartment.xml* that stores just those properties which are available apartments or units that have 2 or more bedrooms. In addition, the structure of *apartment.xml* will be changed from the format represented in *rental.xml* as follows:
  - (1) address will be changed to a single string with comma “,” between street and suburb, suburb and state, and state and a new item country, which should be set to “Australia”.
  - The inclusion of zipcode is optional, but if it is present, it should be between the state and the country, with a space separating the state and the zipcode, and a comma separating the zipcode and the country. No comma is needed between streetNo and street.
  - (2) Attribute *available* will be removed.

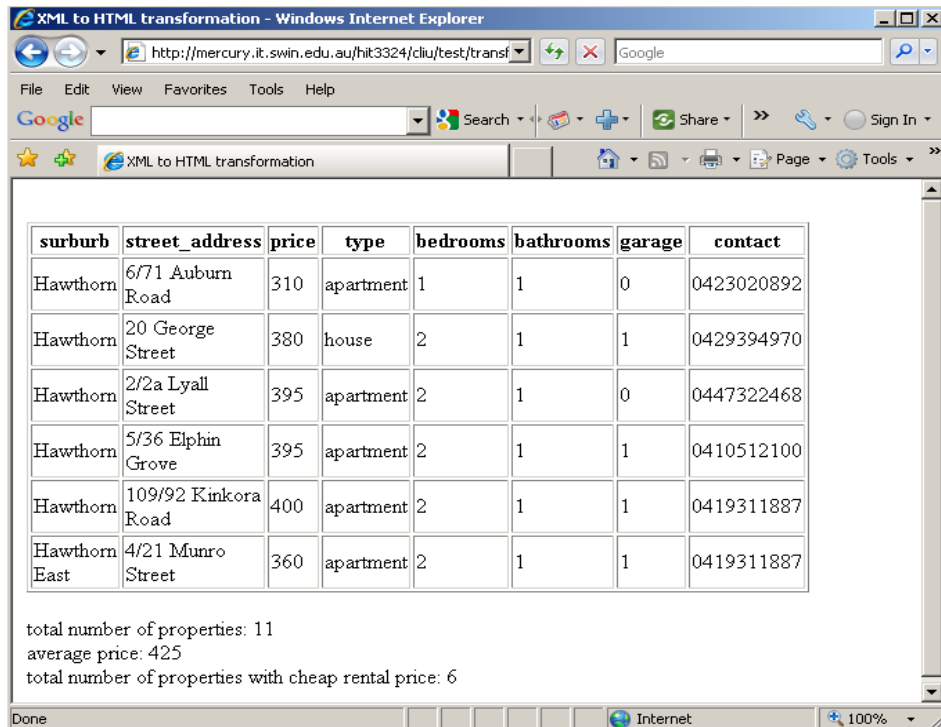
Create *apartment.xml* in the *www/data* directory and display it within the PHP program. Figure 1 shows the structure of *apartment.xml*.



**Figure 1: An example of a generated XML document by *transform1***

- 3) Perform a client-side transformation (*transform2.htm*) on *rental.xml* using an XSLT style sheet called *table.xsl*. The result of the transformation should be an HTML document that will be displayed on the client side. The HTML document is in the form of a table and contains those “cheaper” rental properties that are to be rented at no more than

\$400 per week. The table has eight columns: suburb, street address (streetNo + street), price, type, numberOfBedrooms, numberOfBathrooms, garage, and contact. Below the table, compute and show the total number of rental properties, average price, and total number of those cheaper rental properties (see Figure 2).



suburb	street_address	price	type	bedrooms	bathrooms	garage	contact
Hawthorn	6/71 Auburn Road	310	apartment	1	1	0	0423020892
Hawthorn	20 George Street	380	house	2	1	1	0429394970
Hawthorn	2/2a Lyall Street	395	apartment	2	1	0	0447322468
Hawthorn	5/36 Elphin Grove	395	apartment	2	1	1	0410512100
Hawthorn	109/92 Kinkora Road	400	apartment	2	1	1	0419311887
Hawthorn East	4/21 Munro Street	360	apartment	2	1	1	0419311887

total number of properties: 11  
average price: 425  
total number of properties with cheap rental price: 6

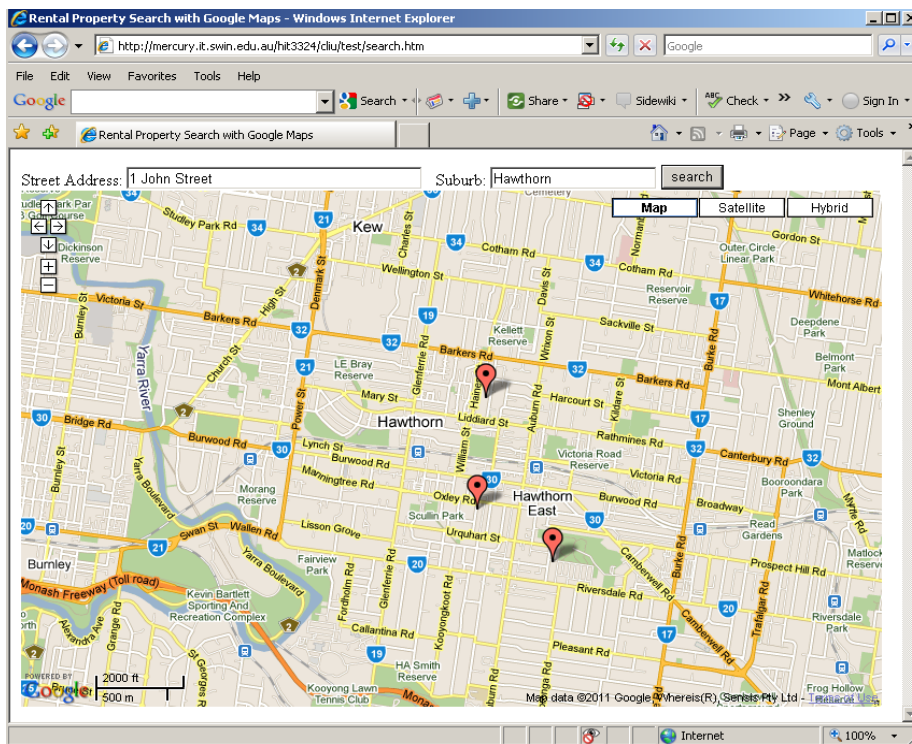
**Figure 2: An example of the generated HTML document by *transform2***

## 2.2 Property Search Service and Map Mashups

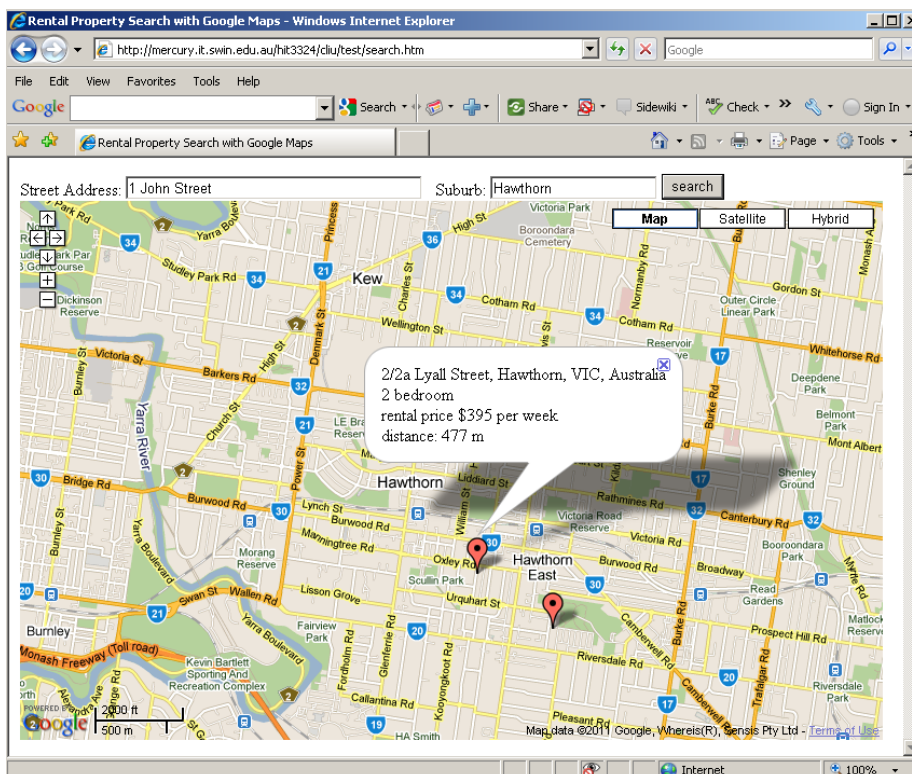
Use the data in *apartment.xml* (use *rental.xml* only if you cannot complete 2.1.2) to create an advanced search service with a Google Map mashup. Your tasks are:

- 1) Design an HTML page (*search.htm*) to take inputs for a search service. Show a text box for a street address (street number + street name) and another text box for a suburb, followed by a search button.
- 2) When the search button is pressed, find three cheapest apartments or units in the suburb (the suburb is included in its address element). In case the rental prices of two or more apartments/units are the same, the one with more rooms will be considered first (see Figure 3).
- 3) In the HTML page, use the Google map API to create a map and show all selected apartments with a marker. When a marker is clicked, the address, price, numberOfBedrooms, contact information, and the distance from the inputted address will be displayed in an information window using the Google map API (see Figure 4). For the distance between the property and the inputted address, either the direct distance (do not worry about any obstacle such as a river in between for this example) or the driving distance is acceptable.

**Note:** you may choose to use API v2. If you use API v3, 4 bonus marks will be given, which can be used to offset the loss of marks in this assignment.



**Figure 3: An example of three cheapest apartments/units**



**Figure 4: An example of an information window**

### 2.3 Upgrading Google map API v2 to v3

As an alternative to 2.2, you may opt to upgrade example4 in Lec9Examples.zip from v2 to v3. You will need to do research on exploring the following link <https://developers.google.com/maps/documentation/javascript/> and map Google map API v2 functions to v3 functions.

### 3 Submission Requirements

You must ensure that all your program files used for the assignment sit in a directory called “Assignment3” (use this name exactly, it is case sensitive and no space between Assignment and 3) under *www/htdocs* directory within your mercury account. This directory should contain no other files and no other sub-directories. However, the *apartment.xml* file transformed from *rental.xml* must be placed under *www/data* directory.

All files required by the assignment description must be submitted as one single ZIP file by using the electronic submission system (ESP). The files should include:

- XHTML files *transform2.htm* and (*search.htm* or *map2.htm*);
- PHP file *transform1.php*
- XSL files *apartment.xsl* and *table.xsl*
- any JavaScript and CSS files, and any additional PHP and XSL files that you use;
- data file *rental.xml*;
- *readme.doc* that includes
  - For 2.2 and 2.3, which you choose to do
  - a list of all the files in the system;
  - brief instructions on how to use the system.

For each submitted file, we require the minimum comments including student information and the main function for the file. After submission, you are not allowed to change any of the submitted files in the Assignment3 directory on your mercury account; time stamps will be checked. **Assignments that fail to follow "submission requirements" will NOT be assessed.**

### 4 Marking Scheme

Work will be assessed based on quality and presentation. The assignment will be marked out of 40 and will contribute 10% towards assessment of the unit.

If you choose to do 2.1 and 2.2:

Assessment item	Marks
Minimum comment, readme.doc and quality of code	3
2.1.1	2
2.1.2	8
2.1.3	8
2.2.1	2
2.2.2	8
2.2.3	9
<b>Total</b>	<b>40</b>

Or if you choose to do 2.1 and 2.3:

Assessment item	Marks
Minimum comment, readme.doc and quality of code	3
2.1.1	2
2.1.2	8
2.1.3	8
2.3.1 map2.html and others	2
2.3.2 load() function, addMarker() function and createMarker() function	8
2.3.3 placeMarkers() function and saveMarker() function	9
<b>Total</b>	40