UNIVERSITY OF TECHNOLOGY SYDNEY

31284 Web Services Development

Autumn Semester 2012 - Assignment 2

Due Date: **Week 13, 8:00 AM, Monday 28 May 2012**

Available points: 100 points

Weighting: 30% of your final grade

Groupwork This assignment must be done as a group of 2.

Part 1 of the assignment is to be done individually by each group member, and will be marked individually. All the remaining parts are to be done as a group.

It is possible to do the whole assignment individually with **permission** from your tutor or subject coordinator (Generally you must convince them that you are a highly competent Java programmer with the ability to get a Distinction in this subject). You will **not** be marked more leniently and will be expected to do the work of 2 students!!

Late penalty:

- Late assignments will be deducted three **percent** per day late (10 points), more than seven days late the assignment will receive zero.
- Failure to demonstrate the assignment will result in a penalty of half marks (-50%) of the assignment.
- Special consideration, for late submission, must be arranged **before** the assignment is due.

Questions?: Refer to the Assignments → Assignment 2 folder on UTSOnline for additional information about this assignment. It is **your** responsibility to read any updates from this folder.

Do not email the subject coordinator questions about this assignment. Ask any questions on the discussion board on UTS online in order to ensure that everyone will see the same question and answer.

Assignment Objectives

This assignment addresses the following objectives from the subject outline.

- 4. Apply concepts of information representation and parsing, in the context of XML and other relevant standards for information interchange
- 5. Describe and evaluate different technology options available for the development of web-based applications
- 6. Develop a small, distributed web-based application based on existing software libraries

Basically this assignment is designed to assess your understanding of the lab exercises

Assignment Requirements

In this assignment you are required to construct a small, web-based application that will assess your ability to work with some or all of the following technologies:

- JavaServer Pages (JSP),
- Extensible Markup Language (XML),
- Extensible Stylesheet Language (XSL),
- XML Schema
- Web services in Java.

The assignment **MUST** be demonstrated during your tutorial in either week 13 or week 14. If you cannot demonstrate during the tutorial, please make separate arrangements with your tutor or lecturer to demonstrate your assignment. The demonstration forms part of your assessment, and you must attend in person.

Failure to demonstrate will result in a penalty of half marks (-50%) of the assignment.

You must use technologies discussed in labs and lectures to solve this assignment. In particular, do not use JavaScript for the main functionality in each part.

Assignment Submission

You will need to demonstrate your application to your tutor during your lab class.
You must submit a single ZIP archive of softcopy versions of:

- The working application
- Documentation (as per part 5)
- ALL source code (but do not include 3rd party libraries)

Other archive formats are **not** accepted – do **NOT** submit **.rar .7zip .tar .tgz** etc files.

You can submit files in a **jar** or **war** file format.
Note that you are limited to a 10MB upload restriction on UTSOnline.
The Zip file must be uploaded to UTSOnline before the due date mentioned on page 1.

Overview

This assignment consists of 5 parts.

Part 1 is divided into two sections labeled (a) and (b). In your group of two students, one student should complete Part 1(a) individually and the other student should complete Part 1(b) individually.

Both students should complete Parts 2, 3, 4 and 5 as a group.

For Part 3 (SOAP Web Services Programming) and Part 5 (Documentation), if only one student in the group contributed effort to completing this part, the marks for these parts can be allocated to one student only (the other student receives zero for the relevant part), but only if there is agreement of both group members that only one student did the work.

Part 1(a) – Student 1 – Basic JSP/XSLT (15 points)

Only **one student** should complete Part 1(a).

Develop a simple 'one page' web application using JSP and XSLT that meets the following requirements.

The web page should display ONE drop-down list with three options as follows. The options may be hard-coded into your application for Part 1(a). The options are:

- **administrative_division**
- **nametype**
- **designation**

Each of these three options corresponds to one of the lookup tables that can be retrieved from the 'lookup' RESTful Web Service (see Appendix for details).

Your JSP should retrieve the results from the 'lookup' RESTful Web Service and then use an XSLT stylesheet to transform the resulting XML into an HTML web page that is appropriately formatted.

In Part 1(a), points will be awarded for:

- Handling user form input, including generation of Web Service URI
- Carrying out XSLT transformation in a JSP
- Quality and style of XSLT coding
- Formatting of resulting HTML page

Part 1(b) – Student 2 – XML Schema (15 points)

Only **one student** should complete Part 1(b).

Create an XML Schema that describes the results of the 'features' Web Service. Details of how to run the 'features' Web Service are in in the Appendix.

You should run some queries on the 'features' Web Service (just by entering URLs directly into your web browser) to view the results. This will show you the elements returned and their basic format. Note that there is an optional element – try searching for features with the name 'Tarraji' and study the results carefully to make sure you have included all elements.

The data we are using comes from the GEOnet Names Server from the US (http://earth-info.nga.mil/gns/html/). You can find more details about the format of the GNS country files data at:

http://earth-info.nga.mil/gns/html/gis_countryfiles.html

Note that we only have a subset of the GNS data, not all fields. Your XML Schema only needs to describe the fields that are returned by the 'features' web service used in this assignment.

You should create your XML Schema to describe the data returned by the 'features' web service as accurately as possible, but you must be careful not to introduce your own extra constraints that are not already present in the data. You should use the GNS country file data format as a guide.

To mark your XML Schema, it will be tested against a variety of XML input files that contain both valid and invalid XML. The points you receive for your schema will be based on how well your schema correctly treats the various input files. The input files used for marking will not be provided in advance. You should make up your own set of XML files for your own testing purposes.

Part 2 - Group Programming (40 points)

Part 2 is not connected to Part 1, i.e. you do not need to use your Part 1 solutions here.

Both students in the group should contribute equally to the development of Part 2 of the assignment.

The goal of Part 2 is to use the RESTful Web Services described in the Appendix to create an interactive web application allowing a user to search for geographic features.

2.1 First you need to create a web page that contains an HTML form with three search fields:

- administrative_division
- designation
- name

In a basic solution, they might all be text fields, but in an ideal solution, the administrative_division and designation will be drop-down selection lists and the name will be a text field.

To use the search function, the user can select one administrative division, and select one designation, and enter a string into the name field. After submitting these values, your application should invoke the 'features' web service described in the Appendix, using the values entered by the user.

The user can choose to leave any of the 3 fields empty, in which case only the fields with values in them should be passed to the web service.

However if the user leaves all 3 fields empty, an error message should be shown and the web service should not be invoked.

After the search has been carried out using the web service, the list of results should be displayed to the user.

As a starting position, 5 points will be awarded for getting a very basic solution working (user enters one or more search values, results are returned and displayed).

Additional points will be awarded according to how many of the following requirements you are able to satisfy, and to what level:

2.1.1 The results should be displayed in an appropriate and visually appealing way. The 'designation', 'administrative division' and 'name type' should be displayed by name only (i.e. no description). The sort key and UFI should not be displayed to the user at all as they are internal information. Your formatting of both the overall web page and the data is important for this criterion. (5 points)

2.1.2 For searching, drop-down lists should be used. These lists should be dynamically populated by using the 'lookup' web service (described in the Appendix) and should **not** be a hard-coded list of values. While for now the web service provides data about Australia, the web service can in future also return data relating to other countries, which is why lookup values must not be hardcoded. (5 points).

2.1.3 The search results should be sorted according to the 'sort_key' element. For bonus points, you should also let the user decide which field to sort by, e.g. sort by designation, or administrative division, or modified date. Even if another field is used as the primary sort key, the results should always be sorted by the 'sort_key' element as a secondary sort level (e.g. sort first by designation, then by sort_key). (5 points)

2.1.4 In the results, sometimes there are entries that have the same UFI (Unique Feature Identifier), but have different names. One name will always have name_type 'N' (Approved), and others will usually have name_type 'V' (Variant). You can gain additional points by grouping these into a single entry in your list of results. i.e. if there are multiple names for the same feature, they should be shown together as a single entry with multiple names, not as separate entries. (5 points)

2.1.5 Your application should be able to paginate the features list since we could retrieve hundreds of features. For this assignment, do NOT use Javascript. You must do this on the server side. Suggested mechanisms:
(simplistic) Next Page button (and display maybe groups of 25?)
(best) Display groups of 1-25, 26-50… or "Page 1, 2, 3, …."
(8 points)

(Part 2 continued …)

2.2 When the user is viewing the list of features, he or she should be able to click on any one of the features in the list and be taken to a new web page. This new web page should contain:

2.2.1 Full information about the feature. This includes all of the information returned by the 'features' web service, but on this page, it should also include the **full descriptions** of the 'administrative_division', 'designation' and 'name_type' fields rather than just their short names. The full description can be retrieved from the 'lookup' web service (see Appendix).

2.2.2 An embedded Google map that shows the location of the chosen feature. The 'location' web service (see Appendix) can be used to retrieve the latitude and longitude of the feature by supplying its UFI.
You can generate a Google Map from latitude and longitude values by using a URL like:

http://maps.google.com/maps?q=-33.890202,151.276199+(Bondi+Beach)

It is up to you to work out how to embed the map within your page. Note that you can achieve all of this using publicly available Google web facilities and it is not a requirement for you to sign up for the Google Developer API.

Part 2.2 is worth up to 7 points depending on how well you implement the requested features.

Part 3 – SOAP Web Services Programming (20 points)

This part of the application deals with calculating the distance between two points.

3.1 Your application should ask the user to select a **source feature** and a **destination feature** and whether they would like results in kilometres or miles. In return, the web service will return the distance between the two features, and the bearing from the source feature.

A SOAP web service is provided that has an operation called getDistance() that will compute the 'straight line' distance between any two points on Earth, and the bearing (the direction you need to be facing from the source point if you want to reach the destination in a straight line).

The getDistance() operation takes 3 parameters:

- **sourceLocation**: a latitude and a longitude of the source
- **destinationLocation**: a latitude and a longitude of the destination
- **units**: either "kilometres" or "miles" (case sensitive)

The latitude and longitude are decimal numbers, with ranges as specified in the WSDL file. All of the latitudes and longitudes returned by the 'location' RESTful Web Service from Part 2 are valid inputs and it is suggested you use these as your inputs.

The getDistance() operation returns 3 results:

- **valid**: a boolean value as to whether the calculation was successful, and whether all the inputs provided were valid inputs.
- **distance**: the distance between the two points in a straight line. This is either in kilometres or miles depending on which units were specified.
- **bearing**: a location on the compass from 0 to 360 degrees indicating the direction you would need to be facing from the source if you were to travel in a straight line to the destination.

Your task is to ask the user to select the source location, destination location and units, invoke the SOAP Web Service, and then display the result to the user.

Note: you do not need to understand how the calculation is performed – all the calculations are done for you by the SOAP Web Service. Your job is just to invoke the web service with the correct parameters.

For Part 3, 15 points are available for implementing the web service call correctly, and an extra 5 points if you are able to integrate this with the application you developed in Part 2. If you cannot integrate Part 3 with Part 2, then you can just ask the user to select from a preset list of features so that you can demonstrate your SOAP invocation is working.

Part 4 – Quality (15 points)

4.1. Your client should be aesthetic and pleasing to use. The generated HTML should be standards compliant (ie: pass validation from w3c http://validator.w3.org/.)
The main characteristics assessed are:
* **completeness** (ie: shows all fields, shows correct data etc)
* **user-friendliness** (i.e.: marker can navigate/use your solution WITH NO HELP FROM YOU).
* The assignment **workflow** should be clear, and should be demonstrable in the classroom. (5 points)

4.2. Your code is **well formatted**, **commented** and easy for the marker to read and understand. To get full points in this part you need to show exceptionally **good coding practice**, and perhaps some innovative techniques. You would show good understanding of design patterns, show evidence of independent research, maybe use recognised design patterns for extra flexibility, and demonstrate that you are generally a top student of the class. (10 points)

Part 5 – Documentation (10 points)

5.1. You should provide brief **documentation** for your assignment - describe how your assignment works and make clear any assumptions needed to run your assignment. This should include some diagrams about your application (such as an overview picture and UML). Basically the information provided should provide enough information to demonstrate that you wrote the assignment and enough information for the marker to install and run your application.

Marking Criteria

This assignment is worth 100 points which are scaled to 30% of your overall subject marks.

Task	Points
1(a) Handling user input Carrying out XSLT transformation Formatting resulting HTML	15 *
1(b) XML Schema correctness as tested against XML input files	
2.1 Search features, display results	5
2.1.1 Well-displayed results	5
2.1.2 Dynamic drop-down lists	5
2.1.3 Sorting (basic or flexible)	5
2.1.4 Handling of variant names	5
2.1.5 Pagination of results	8
2.2 Show descriptions & Google map	7
3.1.1 Invoke SOAP web service	15
3.1.2 Integrate Part 3 with Part 2	5
4.1. Presentation	5
4.2. Coding style	10
5.1 Documentation	10
Total Points	100

* One student completes 1(a) and the other student completes 1(b)

Groupwork assessments

You will be assessed as a team, which means each group member will normally receive the same mark for Parts 2, 3, 4 and 5. If you have trouble with the operation of your group, ask your tutor for advice (preferably ask as a group). If some of the group feel that other members are not contributing the tutor should be informed and a group meeting held to produce a solution. In extreme cases a group member may be asked by the tutor to withdraw from the Subject, do extra work or accept a lower mark. No complaints about group operation will be considered after the assignment has been handed in to the tutor.

Academic Standards

Students are reminded of the principles laid down in the Faculty's Statement of Academic Integrity - Good Practice and Ethics in Informal Assessment found at; <wiki.it.uts.edu.au/start/Academic_Integrity>.

The University's rules regarding academic misconduct can be found at;
<http://www.gsu.uts.edu.au/rules/student/section-16.html >

Assignments in this Subject should be your own original work. The inclusion in assessable work of any material such as code, graphics or essay text obtained from other persons or sources without citation of the source is plagiarism and is a breach of University Rule 16.2.

Asking or paying anyone else to write any part of your assignments is considered cheating. This especially includes using outsourcing websites, and accesses to the subject data are monitored. If you are unsure if your activities (or other student's activities) will be considered cheating, ask your tutor or lecturer for advice.

All text written in assignments must be your own words, except for short, quoted, and **clearly referenced** sections. Text copied from web pages, articles or other sources, and not referenced, will be viewed as plagiarism and forwarded to the Faculty Conduct Committee as misconduct.

Referencing styles may be found at the UTS Library web site <http://www.lib.uts.edu.au/help/referencing>.

Any collaboration with another person should be limited to those described in the "Acceptable Behaviour" section of the Statement of Academic Integrity. Similarly, any group work should be the result of collaboration only within the group. Any infringement by a student will be considered a breach of discipline and will be dealt with in accordance with the Rules and By-Laws of the University.

Students are not to give to or receive from any other persons copies of their assessable work in any form (hard copy or an electronic file). To do so is 'academic misconduct' and is a breach of University Rule 16.2. That is, assisting other students to cheat or to act dishonestly in a submitted assignment.

Accidental submission of another students work as your own is considered to be a breach of University Rule 16.2 in that you are acting dishonestly since you should not have a copy of another student's work.

Specific conditions for assignment 2:

If any parts of your solution are based on existing code that is found on the web, in a book or **generated** by any software application (or taken from any other source), you MUST acknowledge the source as a comment in your code .The only exception to this rule is for code supplied in the subject's lab exercises. Acknowledgement is required even if you start with existing code and make modifications. Failure to acknowledge sources will be regarded as unacceptable behaviour in the context of the Statement of Academic Integrity referenced above.

Using code that belongs to other students currently or formerly enrolled in this subject (i.e. copying or sharing code) is totally unacceptable behaviour and is considered cheating.

The Faculty penalty for proven and serial misconduct of this nature is zero marks for the Subject. For more information go to;

<wiki.it.uts.edu.au/start/Academic_Integrity >.

Appendix: RESTful Web Service descriptions

For this assignment, there are three RESTful Web Services as follows.

1. Lookup

The 'lookup' web service is used to retrieve values from "lookup tables". There are 3 lookup tables used in this application:

- **administrative_division**
- **nametype**
- **designation**

To use the Web Service, the format is:

http://www-student.it.uts.edu.au/~brookes/gns/lookup/*[table]*/*[optional value]*

For example:

- **http://www-student.it.uts.edu.au/~brookes/gns/lookup/administrative_division**
  This will return a list of all administrative divisions.
- **http://www-student.it.uts.edu.au/~brookes/gns/lookup/administrative_division/02**
  This will return only administrative division "02".

The same format applies to "nametype" and "designation" table names.
Both table names and values are case sensitive.

If you provide an invalid lookup table name (or none), an "unknown" table will be returned with no values. If you provide a valid table name but an invalid lookup value, the named table will be returned but with no values.

2. Features

The 'features' web service is used to retrieve a list of geographical features according to parameters provided.

The 'features' RESTful Web Service is at:

http://www-student.it.uts.edu.au/~brookes/gns/features/*[name/value term(s)]*

This web service can take 3 different kinds of query:

- **administrative_division**
- **designation**
- **name**

To perform a query, you need to enter your query word after the keyword above, e.g.

- **http://www-student.it.uts.edu.au/~brookes/gns/features/administrative_division/06**
- **http://www-student.it.uts.edu.au/~brookes/gns/features/designation/ATHF**
- **http://www-student.it.uts.edu.au/~brookes/gns/features/name/Hobart**

You can also use combinations of these queries by concatenating terms, e.g.

http://www-student.it.uts.edu.au/~brookes/gns/features/administrative_division/06/designation/ATHF/name/Hobart

Your query can consist of 1, 2 or 3 terms. The order of terms is not important. Designation codes **are** case sensitive, but name searches are **not** case sensitive.

If you use any keywords other than the 3 mentioned above, they will be ignored (so be careful that you spell them correctly!).

If you don't provide any keywords, nothing will be returned. This is because there are too many records to return all of them at once.

3. Location

The 'location' web service takes a UFI (Unique Feature Identifier) as input, and provides the latitude and longitude of the feature as its result.

The UFI's are the same as the ones returned by the 'features' web service. So to use the location service, you need to first get information from the 'features' service, note the UFI and then use that as input to the location service.

To use the Web Service, the format is:

http://www-student.it.uts.edu.au/~brookes/gns/location/*[ufi]*

For example:

- **http://www-student.it.uts.edu.au/~brookes/gns/location/-1603135**