Individual Coursework: Translated Wise Sayings using RESTful Web Services.

This contributes 15% to your module mark. BB upload by 23:55, Mon 22nd/Nov/2010

The goal is to help you develop and let you demonstrate the following skills:

- using an existing RESTful web service inside a Java program
- designing and describing a RESTful web service remote interface
- implementing a RESTful web service using JAX-RS
- making one web service use another web service
- designing an XML schema for use in a RESTful service
- binding XML documents with Java objects using JAXB

Your tasks, outlined in abstract terms, that will help you to achieve this goal are:

A. Write a Java proxy class for an existing RESTful service offering language translation and use this proxy to complete one missing call in an existing Java application.

B. Design a RESTful service that exposes certain features of a given set of Java classes. The features to be exposed are those used by a simple testing program.

C. Implement your service using JAX-RS and write a modified version of the testing program that uses this service instead of using the classes directly.

D. In your design and implementation, make use of XML representation for one specific composite resource.

The tasks are described in more detail later.

Introduction

It is highly recommended that you read this document very carefully before commencing work. If anything needs clarification, contact the lecturer as soon as possible.

You are encouraged to see the lecturer with your ideas before or during your work to get early feedback. This will help you learn more and consequently increases your chance of getting a good mark.

This document is accompanied by the following Java project archive that should be opened in Eclipse (3.6 for JavaEE developers) as a new Java Project from existing sources:

- `cwk-standalone.zip`:

  In your first coursework task you will be completing one missing portion of code in this project.

  In the subsequent tasks, you will create two new projects, one for a server and one for a client, each similar in functionality to some classes in this original project.

  This project contains the package `wisdoms` that models a database of wise sayings and their rankings. Each wise saying is stored in several supported natural languages (currently English, Spanish and Catalan).

  A saying is inserted in English and will be automatically translated to the other languages. (Nevertheless, this feature is not yet implemented and it will be your task to complete it using an existing RESTful translation service.)

  A simple testing program for the `wisdoms` classes is in package `wisdoms.test`.

Existing simple translation service

The service is available from anywhere on the Internet.

The service provides the following resources and operations:

- a collection of texts
  `http://duck.aston.ac.uk:3000/texts`
  - POST accepting a string representation of a text and responding with a unique numeric text ID for this text
- a text
  `http://duck.aston.ac.uk:3000/texts/{textId}`
  - GET responding with a string representation of the text
- a translated text
  `.../texts/{textId}/transl/{langFrom}:{langTo}`
  - GET responding with a string representation of the translated text
- a collection of supported language pairs
  `http://duck.aston.ac.uk:3000/langPairs`
  - GET responding with a string such as "en:ca ca:en es:fr fr:es". The codes used are the standard locale names, eg "ca" represents Catalan, "es" Spanish, "fr" French and "en" English. (You can click on this URL to open your browser and find out the actual supported language pairs.)
- a language pair (whether supported or not)
  `.../langPairs/{langFrom}:{langTo}`
  - GET responding with either "true" or "false", indicating whether or not this pair is supported by the translator.

Your tasks

1. Complete the missing translation functionality on lines 15 and 16 in file `Wisdom.java` using the above translation service.

   You should not need to add more than 1–3 simple new lines of code to this class besides modifying lines 15 and 16. Most of the code for connecting to the translation service should go to a minimal proxy class which you need to supply. (The proxy does *not* have to support all features of the translation service.) Your proxy should use the Jersey client library.

   You can check whether you have done this task correctly by executing the main method in class `wisdoms.test.Test1`. You should see the texts in (not very good) Spanish and Catalan as well as in English.

2. Design a RESTful service that provides functionality identical to that offered by the `wisdoms` package **and** required by the `wisdoms.test.Test1` program. This means that anything the test needs from the classes in the `wisdoms` package, it should be possible to get from the RESTful service instead of those classes.

   Describe your design in a short document and include it in your solution. (Supported formats: plain text, PDF, ODF, MS Word $\leqslant$2007.)

   Additional requirements:

   (a) All resources that have multiple components have to be communicated using an XML representation and described using an XML schema.

   (b) When an illegal request is made to the server, such as wanting to rate a wisdom using a non-existent identifier, an appropriate error response should be returned with a message similar to the one produced by the standalone application.

3. Implement the service you designed in a new Eclipse *dynamic web project* called `cwk-server`. This can be achieved by copying the classes from the original package `wisdoms` into the new project and modifying them as required by your design and adding JAX-RS annotations. Alternatively, you can add new classes wrapping the original classes and annotate these new classes.

   Some of these classes could be replaced by classes generated from the XML schema(s) you designed. If that is the case, do not edit the generated classes but adjust the other classes to use the generated classes correctly. The `toString` methods and automatic translation from the original hand-written classes can be placed as separate static methods in another class.

4. Implement a proxy for your service and a modified version of `wisdoms.test.Test1` that uses this proxy in a new Eclipse *Java project* called `cwk-client`. You can test your client and server by comparing the output of the client with the output of the standalone testing program.

It is recommended that you work on tasks 3 and 4 in parallel, making more and more of the features available in the service as well as enabled in the test program. This way you can easily test that you have correctly implemented each feature before proceeding to implementing other features.

Assessment scheme

The mark for this coursework will be derived from the demonstrated level of mastering the skills to:

25%	utilise a RESTful service using the Jersey client library (Tasks 1 and 4)
30%	design a RESTful service (Task 2)
25%	implement a RESTful service using JAX-RS (Task 3)
20%	appropriately design and use an XML resource representation (Tasks 2, 3 and 4)

All code will be evaluated in three aspects: correct use of the key concepts (60%), complete compliance with requirements (25%) and good readable coding style, including layout, naming and in-line documentation where appropriate (15%).

Hand-in details

Zip all three of your project folders:

- `cwk-standalone`
- `cwk-server`
- `cwk-client`

in one zip file or three separate zip files and upload these zip file(s) to the BlackBoard assignment tool available at the BlackBoard Module home for CS3250.

Deadline: **midnight Monday 22nd November 2010**

A late submission incurs a deduction. This deduction is 10% of the maximum possible score for every extra working day, unless the year tutor decides to grant exceptions based on extenuating circumstances.