Complementos de Telemática I

Lab Assignment 2: CT1-RESTAuction REST Web Services

Guillermo Vega Gorgojo
November 2010

Abstract

The different sections of this lab assignment deal with the practical use of the concepts reviewed during the course "Complementos de Telemática I" related to REST-style Web Services. The students are expected to develop a very simple bookshop service (and an even simpler client) employing Web technologies and adhering to REST principles. Students should also be able to identify the main difficulties as well as compare this approach with object-oriented middleware technologies.

1 Goals and assignment contents

The goals this assignment aims to achieve are:

- Get an overall understanding of the common steps associated to the development of REST Web Services using a specific Java framework (Restlet) and stressing the differences with respect to the development of centralized applications.
- Assess in practice the main advantages and drawbacks of service-oriented middleware in general, and REST Web Services in particular.
- Compare this approach with object-oriented middleware technologies, assessing their merits and faults.

This assignment does not expect the student to tackle the development of complex distributed applications. It just pretends that the student acquires some basic notions on the usage of the involved technologies so that he can cope with increasingly more difficult concepts and techniques in future formation or professional involvements. Indeed, this assignment consists of the development of the same client/server application that was proposed in assignment A1 with the aim of reflecting on the conceptual differences between service-oriented and object-oriented middleware.

Section 2 introduces the functionality of the applications to be developed. Section 3 describes important advices and guidelines for dealing with the assignment tasks when using the Restlet framework, as well as some important questions that the student should be capable of answering after completing those tasks. Finally, section 5 enumerates some important remarks with respect to procedural and organizational issues.

2 The CT1- RESTAuction application

The distributed application to develop is intended to provide very simple support to the management of an on-line auction company: CT1-RESTAuction.

The application (server-side) will consist of two distributed functional blocks called *ItemManager* and *AuctionManager*. On the one hand, the *ItemManager* maintains a list

of items that can be submitted to an auction. On the other hand, the *AuctionManager* is in charge of the auctions, approving bid requests and checking the closure of auctions. Please take into account that these services can be running in two different hosts.

A user of the distributed application should be allowed to:

- Post an item to the *ItemManager*
- Put an item of the *ItemManager* for sale at an auction
- List the items of the *ItemManager* and their status
- Make a bid for a particular item to the *AuctionManager*
- Check the status of an auction from the *AuctionManager*
- List the auctions maintained by the *AuctionManager*

Important note: the first five functionalities are exactly the same of the assignment A1; the sixth one is new.

3 Development with Restlet

A new Java SE (Standard Edition) 6 project within the Eclipse Development Environment at `balbas.tel.uva.es` should be created with the name A2-AUCTION-REST. It is recommended to reuse as much as possible the code developed for the assignment A1. Moreover, the code developed for the Restlet tutorial (as well as the provided classes `Auction` and `AuctionResource`) should be valid for this second assignment…

3.1 Development steps

1. Develop the implementation of the *AuctionManager* as a REST Web Service. As in A1, it is advisable to use the class `java.util.Hashtable` of the Java API for keeping the information of the auctions updated. This service should be accessed in the following ways:
   a. For creating an auction:
      `HTTP PUT` to the URL
      `http://<hostAM>:<portAM>/auctions/{itemId}`
   b. For making a bid to a particular auction:
      `HTTP POST` to the URL
      `http://<hostAM>:<portAM>/auctions/{itemId}`
   c. For listing the auctions maintained by the *AuctionManager*:
      `HTTP GET` to the URL `http://<hostAM>:<portAM>/auctions`
   d. For checking the status of an auction:
      `HTTP GET` to the URL
      `http://<hostAM>:<portAM>/auctions/{itemId}`

   Note that functionalities **c** and **d** should be accessed with a browser. Thus, it is advisable that the *AuctionManager* serves HTML representations of the auction list (linking to each particular auction) and each auction.

   In contrast, a dedicated client is commanded in development step 4 for accessing the functionality **b,** while functionality **a** is intended for the *ItemManager*. In both two cases, the accepted representations are web forms (as in the tutorial).

2. Develop the implementation of the *ItemManager* as a REST Web Service. As in A1, it is advisable to use the class `java.util.Hashtable` of the Java API for keeping the information of the items updated. This service should be accessed in the following ways:
   a. For posting an item:
      `HTTP POST` to the URL `http://<hostIM>:<portIM>/items`
   b. For putting an item at auction:
      `HTTP POST` to the URL `http://<hostIM>:<portIM>/items/{itemId}`
   c. For listing the items maintained by the *ItemManager*:
      `HTTP GET` to the URL `http://<hostIM>:<portIM>/items`
   d. For checking the status of an item:
      `HTTP GET` to the URL `http://<hostIM>:<portIM>/items/{itemId}`

   As in the previous case, functionalities **c** and **d** should be accessed with a browser, so HTML representations should be served.

   Again, two dedicated clients are commanded in development step 3 for accessing functionalities **a** and **b**. In these two cases, the accepted representations are web forms (as in the tutorial).

   Important note: the *ItemManager* has to access the *AuctionManager* to accomplish some of these tasks.

3. Develop two client applications for accessing the functionality offered by the *ItemManager*. These two applications should be invoked in the following way:
   a. `java a1.rmi.clients.PostItem <name> <seller>`
      This client application should inform the user about the success of the operation and the assigned identifier in case of success
   b. `java a1.rmi.clients.AuctionItem <itemId> <basePrice> <numberOfMinutes>`
      This client application should inform the user about the causes of failure. In addition, it is important to take into account that this client application will only communicate with the *ItemManager*
      Note that `basePrice` is the minimum price of the item in order to be sold, while `numberOfMinutes` refers to the duration of the auction proposal

4. Develop a dedicated client application for making a bid to a particular auction maintained by the *AuctionManager*. This application should be invoked in the following way:
   a. `java a1.rmi.clients.MakeBid <itemId> <offer> <buyer>`
      This client application should inform the user on whether the bid was approved or not

3.2 Questions

These questions can be used as a set of guidelines for identifying important aspects of the assignment that could be considered for inclusion in the written reports (see section 4)

- What were the most difficult aspects identified during the development process?
- Was this development process too different from the centralized case? And from the Java RMI case?
- How is the role of the registry fulfilled in this case?
- Do you see any advantage of using a uniform interface?
- Why it is possible to use an off-the-self web browser as the client?
- Compare the use of Java Exceptions (assignment A1) and HTTP codes (this assignment).
- Distribute in different machines the Catalogue and the Inventory services.
- It is strongly advised to test your clients/servers with those developed by other groups.

4 Procedural and organizational issues

Each student will be assessed by means of a written report and a face-to-face oral review. The written report will have a maximum length of 6 A4 pages (12pt) that will be focused on the problems encountered, as well as on conceptual conclusions (see questions in subsection 3.2). The report will have to be delivered on 10/Jan/2011 and the oral review will take place the same day at the laboratory.