BBC SSE - Elections recruitment exercise

Alfonso Afonso – aafonsoc@gmail.com

General Description

The assignment is based on a set of files (generated by producer – Factory) that must be processed (consumer) and have available the results during the process, control states, duplicates and other issues (Controller). It could be seen from several points of view and different implementations, but mainly I have considered:

- Easy understanding code
- Use of standard tools and frameworks
- Clean concepts and design pattern
- Reusable and easy to be refactored code
- Possibility to implement different classes for producer (factory) and consumer without changes on Controller and Scoreboard concepts
- MVC design

For the project development I have used Java language using Eclipse IDE, and also the use of maven for dependency and project structure. Maven is a plus due to the easy use, flexibility and portability that this tool gives to programmers, besides the simplification of dependencies.

Design concepts and decisions

Primarily the creation of XML structures to convert to Java objects. JaxB tools fit perfectly to this task and is a very common and wide used programming set.

For this reason, I have analyzed some files to define a DTD that match the set of files and then create a WSD file to generate the classes to process the files.

The classes generation is easy to do as maven project, having a jar or the source code available after done it. This project can be done in console with few steps, only needed configuration of basic maven jaxb project files, generate wsd and install.

Second, the basic implementation for the factory and consumer and the relation between them by the controller module.

Initially I thought to create both of them as abstract classes using protected features and some common properties and methods.

Interface was the other alternative, and with this design I can define the methods and basic structure but let the programmer freedom to implement the behaviour and features desired.

As this exercise is to show how to design and program, I have thought that it should be interesting the use of both mechanism, one for factory (interface) and other to consumer (abstract). The decision of let the consumer as abstract class was due to the fact that the JaxB processing and translation is always the same no matter where we will store the data, so it was clear that the best option to code once is an abstract class for this object.

Finally and for the basic data structure and process, I have designed the controller class that is the centre of all the operations. Here I store the partial results already processed. In the constructor is received the factory and the consumer objects to manage the relation between them and calculate the data needed.

Here we control the factory, consumer and scoreboard classes and the information processed itself, using the standard MVC design.

For this class I have considered different implementations, but finally I have decided to do a basic Controller Class and extend it into two other classes: a Runnable class and a Singleton class, both used in Junit test environment:

- Singleton pattern assure us to have only one object created and data is accessible from–to different instances with same results
- Runnable implementation without thread creation inside gives us also the possibility of having several concurrent process reading and processing files and also many clients as we would need consulting processed information

The basic class to store information about parties are two sets of classes, basic class for parties and extended class of parties' scoreboard results with the set of data asked for.

Principal class structures and design concepts

Factory

From abstract class definition, we have a factory that reads all available files from a path and iterate over it when is asked for a product (result)

Also store the result of the process and can be re-loaded, cleaned and retrieved information about the state of one product

Consumer

Given a file, it converts from file-xml-jaxb and checks consistency. After asking about the proper data control by mail (thank you very much for your answer) I decided to let the code to check it, but continuing processing the file if it is xml well formed.

When the file is well formed, data from the constituency is updated in the parties list to be ready for scoreboard processing

Scoreboard

Class that calculate the set of results and return an updated list of all the parties, votes and chairs or only the subset for the scoreboard's format.

I had to implement a "compare" override method to sort the parties Map in a sorted List and do an easy linear calculation of the results (first three parties and sum for others).

Controller

The main class for this application, it is in charge of use the factory to get elements to be processed and analyze them with the consumer, taking the updated data and prepared to show it (connect to View module).

The implementation consists in a basic class (fully operative) and two extended classes that show other way to work, a Singleton and Runnable (Threads) class.

This class receive the factory and consumer in the constructor instead of creating itself to allow having, for example, one factory based on DB and one consumer in Map (Collections), so is more flexible and the programmer does not need to change code to reuse it with the desired factory and/or consumer.

Scoreboard Behaviours implemented

The program can be easily run from eclipse junit test showing two sets of test, singleton and threads.

The behaviours described in the assignment has been all done:

- auto process files in the application (after running)
- validation of files and inform to previous class (and also to factory)
- control of files, process and information
- show results as described (three first plus “others” sum) with percentages and the “winner” mark when one party has more than the half of the chairs

Improved features

After having the model and controller developed and tested, the following steps that I consider to this project are:

- use database persistence of data, so you can store and process files only once and access to results using different tools and methods
- creation of simple REST object to give a XML – JSON scoreboard results

I think that the first step must be include some database system (sqlite, mysql or postgresql) to store either parties results and processed files.

Having this improvement you can have full access to data from any place with data real persistence (you do not have to calculate twice the files), and even from different languages, methods or environments.

The second step will be prepare a basic REST set of tools to show how could be used our project and show the features implemented.

<u>Persistence DBMS (sqlite, mysql or postgresql)</u>

To prepare the program to persist using a database system, what I have done is:

- create a subset of tools to access database. As I am not sure what resources will be available to test the project, I decided to create a Sqlite database that shows the same behaviour than other more complex and advanced RDBMS as MySql or PostgreSql.

- create extended class from AbstractResultConsumer to store data after process any file. We only have to override the updatePartiesList method to store data into database.

- create a new class from InterfaceResultFactory to store states and results for any processed file (and also to store list of available files).

All this features can be done either with JDBC or Hibernate database control, creating some basic classes to manage SQL access.

I decided to access directly to JDBC with SQL language due to I feel very comfortable when I am using SQL language to manage data, although Hibernate objects with JPA annotations would be a great solution too.

<u>Important notice</u>

Due to the features of SqLite the multi-thread test can not be performed properly, having issues with locks and permissions. Using a multiuser environment like previously named we can use the threads with a RDBMS without any problem.

Controller to View access: REST service

Having the controller and model completed and with different ways to store and manipulate data, I took the last step to be able to show the information processed in a multi concurrent environment with proper data structures.

For this reason, I have though that it would be a good idea to develop a small REST project (also with maven structure and JaxB xml control), so what I have done is:

- prepare a dtd to store the scoreboard (the set of results asked in the assignment)
- generate JaxB classes
- create a new project with maven which uses the main project (constituencyResults is the main project with controller and model, this new project is called constituencyResultsREST as part of the controller to view part of the project)
- generate the basic interface using Jersey and the singleton ConstituencyResultsSingleton class to process

I test the REST project under Eclipse with Tomcat 7, using version 1.6 for runtime environment and jersey version 3.0, so you can have dynamic modules load feature, and the process to test is quite simple:

- put the files in a path accessible to the server (I put them in my home root folder, is a parameter that you can send to factory constructor)
- launch the module in server, directly or under eclipse
- call init (you can test to call other but while there is no object created, you will have a empty result or a message alert)
- process files, you can call either procfile(process only one file) or procall (process all files)
- show results in txt or xml (the normal use, txt was only for testing purpose) by scoreboartxt or scoreboarxml

The root path is http://localhost:8080/constituencyResultsREST/resources/elections/ and the method we would like to do (init, clear, procfile, procall, scoreboardtxt or scoreboardxml). Without path shows a very simple about message.

Files, documents and material delivered

With this document explaining the implementation of this assignment, you can find:

- main maven project with all the code inside and a set of test (JUnit) already prepared to show the functionality of the system
- second maven project with REST interface, linked by maven dependencies to the main project, with the basic implementation for REST service (init, clear, procfile, procall, scoreboardtxt and scoreboardxml)

In the main project you can find also some log/process files and the normal structure for the maven project and tools used (log properties, sqlite db, documentation).