Monash University
Faculty of Information Technology

FIT3077
Software Engineering: Architecture and Design

Assignment 2 – Semester 1, 2017
Team Software Architecture Assignment
Weather Monitor

Outline

Aim

We have discussed a variety of important architecture and design principles in this unit. **The aim of this assignment is for you apply these principles and patterns in actual code.** This will give you the opportunity to reinforce your understanding of these ideas, and to demonstrate that understanding to the lecturer. This is a design assignment: **simply producing a program that implements the required functionality is not enough to pass**.

Teams

This is a team project. You will work in pairs. You should already have seen the Moodle News Forum message about registering your team with the unit coordinator and your lecturer via email. All teams will use a GitHub repository to manage their code and other documents. Assignment submission will be done by taking a snapshot of your GitHub repository.

Teams will be interviewed about their work. Both team members will be required to demonstrate understanding of all parts of both the software design and the code.

Stages

This assignment has two stages. The Stage One requirements are in this document. The Stage Two requirements will be released in the Week 10. In Stage Two you will extend the functionality of the system you have created for Stage One. It is important that you keep this in mind when designing your system for Stage One.

Programming Language

This assignment is language-neutral. You may choose to use any object-oriented language you like. You can use a language that you already know, or choose to take this opportunity to learn a new one. The only requirements are:

- You must be able to demonstrate your program to unit staff. You can do this on a machine in the Monash computer laboratories, or on a laptop computer.
    - If you intend to demonstrate on a laptop, make sure that you have access to the Monash wireless network enabled on that machine.
- Your program will need to act as a client to a web service using the SOAP protocol. SOAP libraries and tools exist for many languages.
- Your program must have a graphical user interface.

Web Services

In this project, you will use a *web service*. The *World Wide Web Consortium* (W3C) defines a web service as “a software system designed to support interoperable Machine to Machine interaction over a network.”[1] Web services allow you to create objects in your program that act as interfaces to services running on other machines on the internet. Once you have set up the service, you can use the service object as if it were just like any other part of your program – the fact that calls to its operations actually result in messages getting sent across the internet is hidden.

There are two main protocols for defining and using web services: SOAP and REST. The web service you will use for Stage One uses the SOAP for communication between the client and the server. SOAP used to stand for *Simple Object Access Protocol*, but as of version 1.2 this acronym was dropped. SOAP is an XML-based communication protocol that often uses HTTP as its transport method. Again, you should **not** need to know any of these details to use a web service – the libraries and tools of your chosen language should hide all this from you.

The properties of a SOAP web service are specified using the *Web Service Description Language* (WSDL). The WSDL is an XML-based language. The WSDL file provides definitions of

- How to connect to the service
- The operations provided by the service
- The input and output messages for each operation
- The data types used in the messages

You should **not** need to look into the details of WSDL files at all for this project, but you will need to use one to tell your program about the service that you are using.

Stage One Requirements

The Weather Monitor Application

You are going to design and build a Weather Monitor Application. The requirements for your Stage One system are:

- The user can choose to view the temperature and/or rainfall at locations selected from those supported by the MelbourneWeather2 web service[2]. For each location, a separate monitor must be created. Your program should check the web service every five minutes, and update the monitor(s) as needed. The numerical temperature and/or rainfall data should be shown in the monitors for the locations being viewed.
- The user can choose to view the weather for more than one location simultaneously.
- The user can choose to stop viewing the weather for a location.

The GUI design for this system is up to you.

You must minimize the network traffic generated by your system: don’t access the web service more than is absolutely necessary.

When designing your system, you should bear in mind that extensions to this functionality are expected in Stage Two. In particular:

- You might be required to have more than one kind of monitor for the same location at the same time.
- You might have to display the data in a non-numerical format.
- You might be required to support a new web service that supplies the same sort of data, but not necessarily in the same way.

[1] http://www.w3.org/TR/ws-gloss/

[2] See next section

The MelbourneWeather2 Web Service

The WSDL file for the simple web service you will use is at:

http://viper.infotech.monash.edu.au:8180/axis2/services/MelbourneWeather2?wsdl

This describes a web service called MelbourneWeather, that provides three operations:

- `getLocations`
- `getTemperature`
- `getRainfall`

`getLocations` returns an array of strings, each of which is the location of a weather station in the Melbourne area.

`getTemperature` takes a location string as an argument, and returns an array of two strings. The first element of the array is a timestamp. It gives the time that the weather data was last updated. The second element gives the current temperature at that location in degrees Celsius. Sometimes no data is available from the weather station, in which case "-" is returned.

`getRainfall` takes a location string as an argument, and returns an array of two strings. The first element of the array is a timestamp. It gives the time that the weather data was last updated. The second element gives the rainfall since 9:00am on the current day at that location, in mm. Sometimes no data is available from the weather station, in which case "-" is returned.

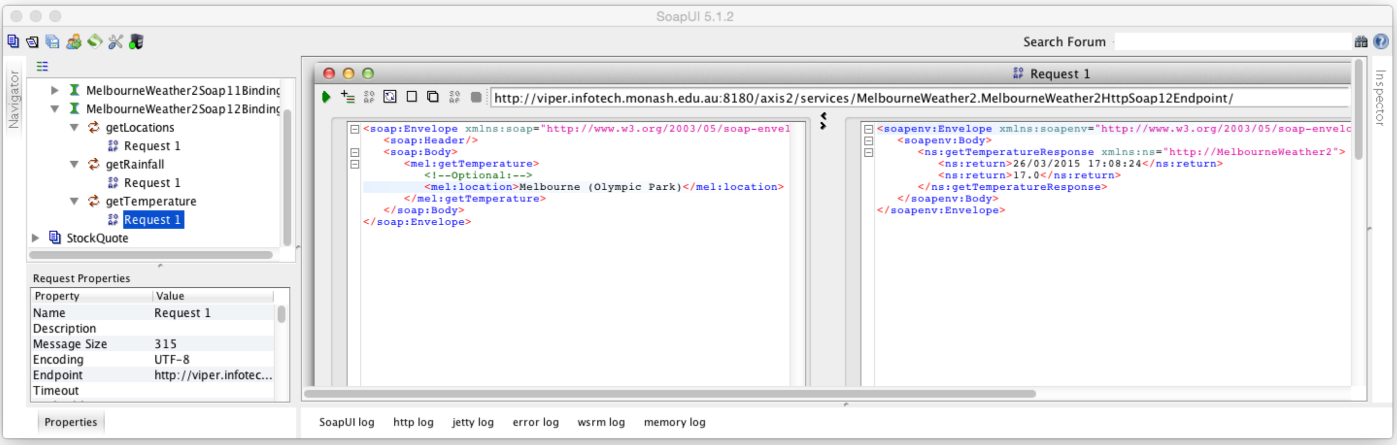
The temperature and rainfall data are updated every ten minutes.

Technical Note

The MelbourneWeather2 service runs on the machine viper.infotech.monash.edu.au. It is accessible from both outside Monash and within. It has been tested using Java (using Eclipse Kepler and Neon with Apache Axis2), and also via SoapUI (see below). Sample code showing basic use of the service will be made available on the Moodle site. This is a fairly old web service: working out how to adapt legacy services for incorporation into new systems is a common architecture and design challenge.

soapUI

If you are interested in seeing what SOAP messages actually look like, you can use a tool such as the free SoapUI: http://www.soapui.org/. This lets you provide a WSDL file and then test a web service, showing the SOAP XML request and response messages in full, e.g.



http://viper.infotech.monash.edu.au:8180/axis2/services/MelbourneWeather2.MelbourneWeather2HttpSoap12Endpoint/

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envel
   <soap:Header/>
   <soap:Body>
      <mel:getTemperature>
         <!--Optional:-->
         <mel:location>Melbourne (Olympic Park)</mel:location>
      </mel:getTemperature>
   </soap:Body>
</soap:Envelope>
```

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelo
   <soapenv:Body>
      <ns:getTemperatureResponse xmlns:ns="http://MelbourneWeather2">
         <ns:return>26/03/2015 17:08:24</ns:return>
         <ns:return>17.0</ns:return>
      </ns:getTemperatureResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Request Properties

Property	Value
Name	Request 1
Description	
Message Size	315
Encoding	UTF-8
Endpoint	http://viper.infotec...
Timeout	

Stage One Deliverables: 20 marks

The deadline for the Stage One Deliverables is 23:59pm, Sunday, April 30th (start of Week 9). You are required to provide, via your GitHub repository, the following:

- UML design documents for your system. Include whichever UML diagrams you think are needed to explain your design. If you are unsure of whether a diagram is needed, ask the lecturer. **These must be in PDF format. Do not** provide only files for a drawing or modelling program (e.g. do not provide only .vpp files for Visual Paradigm). **(10 marks)**
- Your source code. Object files (e.g. .o files, .class files, etc., depending on the language) and executables should **not** be placed under version control in your repository. **(8 marks)**
- Git log (no need to submit this. It happens automatically.) **(2 marks)**

There will be no submission per se. The unit staff will take a snapshot of your Git repository at the due time. This snapshot will be your team's submission. You will also be required to demonstrate your system to your tutor during practice class time.

Designs and code will be assessed for quality, **including extensibility and appropriate use of design principles and patterns**, and use of appropriate coding standards. Completeness of code functionality with respect to the specified requirements will also be assessed, but **completeness of functionality is not enough to pass by itself**. Design quality is the most important criterion.

Your Git log will be inspected to ensure that both team members have contributed significantly to the project, and that all commits have meaningful messages associated with them.

Stage Two Deliverables: 80 marks

The deadline for the Stage Two Deliverables is 23:59pm, Sunday, May 21st (start of Week 12). The deliverables are:

- UML design documents for your system. **These must be in PDF format.** **(25 marks)**
- Your source code. **(20 marks)**
- Git log (no need to submit this. It happens automatically.) **(5 marks)**
- Written explanation of the design principles and patterns that you have applied in the design of your system. You must explain the reasoning behind your class model in terms of the principles and patterns described in lectures, or in other design texts. All sources must be properly cited. This should be no longer than two A4 pages. **(15 marks)**
- System demonstration and interview. **(15 marks)**

Team interviews and demonstrations will take place in Week 12.

Allocation of Marks

This assignment is worth 40% of the marks for the unit. There are 100 marks available, allocated to the two stages as shown above.

The assignment is to be completed by teams of two students. It is acceptable to obtain help and input from other students, but this should be acknowledged directly in your submission. In the event that a submission has been copied from the work of another student, all students submitting copied work will receive zero marks for the assignment.

Extensions

Extensions will only be granted if students are able to provide medical certificates or other supporting documentation. Extensions will not be granted on the grounds that the student has "too much work", or has other assignments due. If an extension is needed you must contact the lecturer as soon as possible, **before** the submission date. It will not be possible to apply for an extension after the submission date.