Service-Oriented Architecture Module: *Pre-study*

Paul Fremantle, paul@fremantle.org, June 2016

Since the World-Wide Web opened up distributed computing for a huge range of users and applications, programming models to support that kind of distribution have been multiplying rapidly. The range of technologies involved in delivering such software can be bewildering, but some relatively straightforward principles for service orientation are emerging.

In this course we will study enough of the technologies of web services (SOAP and REST) to enable us to implement, call, and intermediate simple services, but also try to engage with the broader issues of service orientation: how to design and engineer services and web APIs, orchestration, security and management.

The textbook ***Service-Oriented Architecture: Concepts, Technology and Design*** *by Thomas Erl* is excellent in where it goes, but it is a little dated. Unfortunately, there isn't a single book that covers everything I'd like. This is definitely the best book for the overall vision as well as the WS-* side of SOA. There are other excellent books that cover RESTful concepts but don't have the same overall introduction.

An ideal preparation for the course would be to read the whole of Chapters 1-3.

To balance out the very WS-* focused approach of that text book, I suggest you also do some pre-reading around RESTful design approaches.

I recommend the following free resources:

- http://www.infoq.com/articles/rest-introduction
- http://www.infoq.com/articles/webber-rest-workflow/

The **practicals** will be managed with open source tools using the following technologies:

- **The Linux Command Line shell** (bash):
  If you have never used a Linux Command Shell, please read:
  http://linuxcommand.org/lc3_learning_the_shell.php
- **XML** including XML namespaces and XML schema
- **JSON**
  JSON is a very simple model for data interchange.
  The following article is a reasonably good introduction.
  http://msdn.microsoft.com/en-us/library/bb299886.aspx
- **Java**
  Some of the practical tasks will rely on the Java programming language.

We will make them as accessible as possible for those who are not experts in the language, but if you have an opportunity to review some of the details of Java before the course, please do. We will use the Eclipse tool for developing Java code.

- **Other languages:**
  Because SOA is inherently a language neutral system, we will be using multiple languages. In particular we will use Python and JavaScript as well as Java. If you have never used these before please take a look at the Python tutorial: https://docs.python.org/2/tutorial/ (especially sections 1-5

Pre-exercise

In addition to the reading, I'd like you to do a simple exercise.

The objective of this exercise is to help you understand your current approach to designing a computer system. The aim of this is to lay a foundation for future discussions during the week.

This is not a test.

There are no right or wrong answers to this exercise and the results are not handed in. We will have a short discussion about the exercise during the course.

I'd like you to spend around 30 minutes on this. Spend no more than 45 minutes.

Scenario

You have been offered the job of CTO of a startup. The startup's aim is to make it easier for audiences to find live music events that they like. To this aim, you need to support the following functions:

- Allowing users to sign up to your website, either directly or via facebook
- Letting bands and performers to sign up and post their gig listings, including dates and locations.
- A system whereby bands can directly link their event listings from Facebook and Songkick (as well as other band listing services) into this site.
- Expose APIs for other websites to post or read listings automatically
- Letting fans follow bands
- Allowing a fan to get notifications of when gigs are coming up within a specified geographic radius of their location

- Linking to existing ticketing systems so that this website can sell tickets directly.
- Linking to a payment system to support accepting payment for tickets
- Recommending gigs they might like based on their previous gig bookings, facebook likes and/or streaming playlists.

This is not about designing a real system in half an hour!

Instead, please take the time to think about the core technologies you might use to build this system, the design approach you might take. If you have experience with using SOA, would you use it for this system? If you don't have experience with SOA, what technologies do you know that would allow you to build this? What are the challenges in building this?

Any work you create for this exercise is just for your own use, but you might consider drawing some quick and dirty versions of: interaction diagrams, system architecture diagrams, models, schemas or other diagrams that might help you.

I hope this "baseline design approach" helps you later in the week as we revisit these types of design decisions.