<< Cover Page>>Duc Duc

Hi and welcome to the Zero to Cognitive Series. This tutorial is designed to introduce you to building cognitive applications using the IBM Bluemix Cloud. We will use the IBM Cognitive Solution Advisor as our foundation for this tutorial. This is Chapter 1.

<< Page 2 >>

The intent of this tutorial series is to enable anyone to get started easily with IBM Bluemix and the IBM Watson services. Both of these can be freely accessed at <https://console.ng.bluemix.net/registration/>

I’ve been asked: ‘With all the sample code out there, why this tutorial?’ It’s a really good question. The available sample code, which I use regularly, is written for developers – that is, people who already understand the Bluemix environment, are current in one or more programming languages, and are looking for how-to examples. Much of the code already in place focuses on how to access a service, but does not include how to use that service in a compelling way. This tutorial has been designed to make it very easy to get started and also include creating a more compelling user experience.

<<Page 3>>

This tutorial is currently planned as 12 modules. I expect that we will do many more than that as the available IBM Watson services continue to evolve and grow. The first half dozen lay a foundation for a web app which will exercise some of the core Watson services and create an interactive environment for future development. The next half dozen expand on this foundation and integrate Watson Alchemy and use noSQL databases to store important information.

From there we’ll temporarily move away from Watson and look at handing complex and hybrid environments in IBM Bluemix. Then we’ll circle back and extend our foundation app with a custom inference engine.

<<Page 4>>

Our reference application, The IBM Cognitive Solution Advisor, was initially conceived as part of IBM’s Cognitive Build challenge in 1Q 2016. We wanted to create a ‘find’ service as opposed to one more ‘search’ engine and had the idea to base this on natural language conversation. We thought that if we could create an app which allowed anyone to describe what they observed or were interested in, then the process of finding IBM solutions would be simple, intuitive and easy. This storyboard describes that high level interaction.

<<Page 5>>

A System Context diagram is used to highlight the core systems and services with which any application will interact. In this diagram, we can see that we initially identified the need for 4 Watson services along with access to 3 different IBM internal data repositories. This diagram was used on the following page to begin to establish an overall architecture for the application.

<<Page 6>>

Perhaps the first question we should ask is: ‘Do we really need architecture in an Agile, Dev/Ops environment?’ If the app is small enough and there’s only one person working on it, the answer might be “No”. But this is not planned to be just a tiny little app and the first month of development on it was done using a crowd-sourcing approach. So why architecture? Very simply, our architecture enabled us to accomplish 2 important objectives simultaneously:

1. We could separate work across 6 different teams and allow all of them to work independently
2. We could streamline integration by establishing ownership of key interfaces and stabilize those interfaces at the beginning of the project

Along with the architecture shown on this page, we also established a small set of core governing principles for our work:

1. Separation of Concerns & Loose Coupling
2. Restful Services
3. Mobile code is present to drive UX

Separation of Concerns and Loose Coupling meant that we were able to and required to keep logically aligned sections of code together and, at the same time, present a ‘black box’ to other teams. The black box approach meant that any team could use any available service without needing to be concerned with how that service worked.

Restful Services is a logical extension of the previous principle and establishes how any server-side service will be accessed.

Mobile code is present to drive UX is a current best practice in the design and development of Systems of Engagement (mobile) applications. It enables the mobile development team to focus just on creating a compelling user experience and requires that all business logic be managed via restful services. This makes the business logic available to any other application and simplifies design, coding and application delivery cycles. It makes the code development exceedingly Agile.

The core components of the IBM Cognitive Solution Advisor are the

1. Mobile UX code and Watson Speech
2. Security and Session Management
3. Watson NLC and Dialog Management
4. The Inference Engine

We will use this as our basis to build a web version of this application. Let’s look at a demonstration.

<<go to live demo>>

Once we’ve logged in to The Advisor, we can hear Watson ask us what we want to work on. When I press the microphone button, Watson Speech to Text is activated and our voice is turned into text when I tell The Advisor that ‘My client is interested in improving the quality of their marketing campaigns’.

The Advisor responds by asking me if it has correctly determined my client’s industry, which is pretty cool since I never specified an industry. If its correct, I say yes, otherwise I say No. In this case its incorrect so I’ll say No and then tell it CPG. The Advisor then verifies that it has the correct Role and will then ask if I want to provide any additional information. When I tell it “no” to the last question, The Advisor will now go find solutions.

Over the next few sessions, we will recreate a simplified web version of this application.

The next tutorial in the series will be on setting up Bluemix and getting the requisite code onto your workstation so that you can begin creating your own unique Cognitive applications.