International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Web Service Application to Access Climate Data from Cloud using Spring Framework

Vijayshree[1], Dr. Rekha Patil[2]

[1]Department of Computer Science and Engineering M.Tech (CSE), Poojya Doddappa Appa College of Engineering, Gulbarga, Karnataka, India

[2]Department of Computer Science and Engineering, Poojya Doddappa Appa College of Engineering, Gulbarga, Karnataka

Abstract: ***The world has full of valuable information because of the rapid development of internet, World Wide Web and e-commerce application. It is a huge challenge to fetch this valuable information which is distributed in different parts of the world and in a different format. The field of application where sensors are used is expanding extensively fast. Sensors provide information about various aspects of the real world. Sensors are used for gathering environmental information such as changes in temperature, air pressure, and water quality. The important task is to analyze the large amount of multidimensional data generated by diverse sensors and interoperability among various system which handles such complex data. These sensors are distributed geographically and their networks do not take context information into account due to which they provide heterogeneous interfaces to access the data. To handle these issues we need to give greater importance to the interoperability of the data acquired from heterogeneous and distributed network of sensors.In this paper, We are proposing an architecture where the sensor is automated using Jenkins Continuous Integration server and generate the climate data information from the cloud and store in the NoSQL Mongo database and expose the climate data using Spring RESTful web service over the web and visualize the climate data to users using Spring Model View Controller framework. It also provides the performance benchmarking of REST and SOAP Web services.***

Keywords: Sensors, Cloud Web Services, RESTful Web Services, Jenkins Continuous Integration, NoSQL, Mongo DB, Spring MVC, Spring RESTful web services

1. Introduction

The explosion of the web, and now web 2.0, it is only now that we are beginning to see the automatic interaction between different web sites, and between web sites and client applications, web sites and business databases, as part of more a global interconnection.

On the web, data is moving faster than we can browse it. Hence, there is a strong demand for everyone that can find, track and monitor information coming from diverse sources.The rapidly emerging web services technology has the potential to change the way the Internet works for businesses. While currently web services work well for business-to-consumer applications, they don't for business-to-business applications. RESTful Web services should be able to address this issue.

The motivation behind RESTful web services is what drives the progress of technology to make complex things simpler. RESTful Web services quickly becoming the preferred technology for building arbitrary applications that communicate over the network. REST fully leverages protocols and standards that power the World Wide Web and is simpler than traditional SOAP-based web services. With the emergence of cloud-computing and the growing interest for web hosted applications, REST-based technologies can help both in the development of rich user interface clients calling into remote servers; and in the development of actual servers for manipulating data structures in a client application (written in any language) or directly in the browser.

The study mainly focused on exposing the climate data coming from sensors as REST API using Spring RESTful web services and development of client application which consumes the REST API and visualizes the Climate data using Spring MVC Framework.

The study also extended in usage of NoSQL Mongo DB which is an effective way to store large amount of climate data. In addition, we used Jenkins Continuous Integration which interacts with cloud tofetch the data and prove that CI is best for automation framework.

The proposed system makes use of internet data packet and JSON messages are realized by RESTful web service.There will be three major components residing on different tiers, which communicates with each other and to external services using JSON messages to realize the complete system.

The first step on implementation would be developing a sensor application which is responsible to establish connection between sensor and web service.The next step is to develop a client application, which can be accessed via URL. It is a user intractable application and responds to user's request by fetching data from the RESTful web server.The final step is to develop a sensor web service, which is the core of our system, which is responsible to process, coordinate and manage sensor data among sensor devices and the client application. It will exchange JSON messages to realize the data communication between the sensor application and client application.

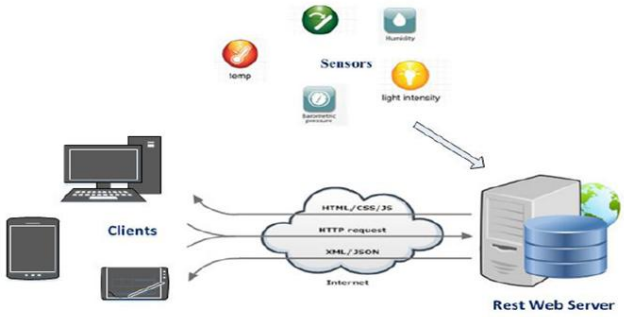

Figure1: RESTful Web Services

The paper is organized, section 1 discusses the introduction, and section 2related works. Section 3 methodology, architecture and system design Section 4, result. Finally, section 5 presents some concluding remark.

2. Related work

"A Sensor Web Middleware with StatefulServices for Heterogeneous Sensor Networks": **T. Kobialka,R. Buyya, etal[1]** has proposed Web Service Resource Framework (WSRF) mechanisms into the core services implementation of the NICTA Open Sensor Web Architecture (NOSA). NOSA is a suite of middleware services for sensor network applications which are built upon the Open GIS Consortiums Sensor Web Enablement standard.The NICTA Open Sensor Web Architecture is built upon a uniform set of operations and standard sensor data representations as defined by the Open Geospatial Consortium (OGC). The OGC is a geospatial standards authority that has defined a Sensor Web Enablement (SWE) method which includes specifications of interfaces, protocols and encodings that enable discovering, accessing and obtaining sensor data as well as sensor-processing services.

"Architecture for implementing geospatial web services using SOA and NoSQL solutions" -P. Amirian, A. Basiri, A. Winstanley[2]The proposed work assesses geospatial services through a so called OGC services framework and proposed an architecture for implementing geospatial web services.The first approach is to use the specifications published and managed by Open Geospatial Consortium (OGC).The second approach is to use web services technologies and use the standard messaging and standard interface definition mechanisms**.** In this paper in order to provide access to geospatial resources in efficient and standard manner there are generally two issues: huge volume of geospatial data and limited capabilities of OGC services to provide self-sufficient messages containing geospatial data. The proposed solution provides geospatial resources through interoperable services which can be consumed using various kinds of message exchange patterns for different kinds of clients**.**

"Service Oriented Architecture:SOA strategy, Methodology and Technology"- J. Lawler, B. Hawel[3] The proposed works for an architecture for implementing web services using SOA (SOAP, WSDL, UDDI)The concept of SOA solves the traditional problem of tight coupling and Web Services are the major technology for implementing SOA

"Using Service-Oriented Architectureand Component-Based Development to Build Web Service Applications."- Brown. A., Johnston,S., Kelly, K[4]The proposed work have written a Rational Software White Paper and concluded that 1. Services models can be derived from existing component-based models. 2. Common patterns and designs can be applied to assist in the transformation to service-oriented models**.**The system as a whole is designed and implemented as a set of interactions among these services. Exposing functionality as services is the key to flexibility. This allows other pieces of functionality (perhaps themselves implemented as services) to make use of other services in a natural way regardless of their physical location. A system evolves through the addition of new services. The resulting service-oriented architecture (SOA) defines the services of which the system is composed, describes the interactions that occur among the services to realize certain behavior, and maps the services into one or more implementations in specific technologies.

"The Road to Enterprise Application Integration (EAI)"- ZaighamMahmood[5] the proposed work for EAI integration based on SOA.Enterprises have invested heavily in large-scale applications software to run their services and business functions. The infrastructure used is often heterogeneous across a number of platforms, operating systems and languages and, thus, there is a huge duplication of functionality and services resulting in a waste of valuable resources and poor response times. Increasingly, the business and IT managers are being asked to deliver improved functionality while leveraging existing IT investment as well as provide flexibility and on-demand services. In this context, Service Oriented Architecture (SOA) is emerging as an attractive architectural style for developing and integrating enterprise applications. SOA promises a better alignment of IT with business, seamless integration of business applications and reduced costs of development and maintenance. Evidence suggests that large enterprises are moving towards this new paradigm.

"Restful web services: A solution fordistributed data integration"- J. Meng, S. Mei, and Z. Yan[6]
has proposed work have a solution for distributed data integration using RESTful web services.The core principle of RESTful Web Services is resource which is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time. Representations are any useful information about the states of the resources. They are usually a sequence of bytes, plus representation metadata to describe those bytes. These advantages are suitable for solving the problems of Internet-scale distributed data integration. Traditional Web Services have more mature extension standards of WS-* protocol stack and good support of development tools to meet the needs of QoS and complex heterogeneous hardware and software system environment. So it is more suitable for enterprise computing. But we believe that both kinds of Web Services have complementary strengths.

"REST and SOAP: When Should I Use Each (or Both)"- Rozlog. M[7]has proposed work by taking an example and demonstrated the two approaches for interfacing to the

Web with web services, namely SOAP (Simple Object Access Protocol) and REST (Representational State Transfer). Both approaches work, both have advantages and disadvantages to interfacing to Web services, but it is up to the web developer to make the decision of which approach may be best for each particular case.The approach is very simple to understand and can be executed on really any client or server that has TP/HTTPS support. The command can execute using the HTTP Get method. So developers that use this approach, cite the ease of development, use of the existing web infrastructure, and little learning overhead as key advantages to the style.

" Migration ofSOAP-based Services to RESTful Services"-B. Upadhyaya, Ying Zou, Hua Xiao, J. Ng, and A. Lau[8]has proposed a case study on a set of publicly available SOAP-based Web services. The results of the case study show that their approach can achieve high accuracy of identifying RESTful services from the interfaces of SOAP-based services and can improve the performance for invoking Web services after SOAP-based services are migrated to RESTful services.

"Implementing REST Web Services"-Hao[9] the proposed work have some best practices and guidelines for implementing RESTful web services. With other styles of web services, REST is easy to implement and has many highly desirable architectural properties: scalability, performance, security, reliability, and extensibility. Those characteristics fit nicely with the modern business environment, which commands technical solutions just as adoptive and agile as the business itself.

"RESTful Web Services Development Checklist[10] The proposed work has discussed on rich functionalities of REST and HTTP. He proposed the IDE and checklist to develop the REST APIs in easy and fast manner. RESTful Web service developers can have well-behaved service implementations up and running in short order. REST's relative simplicity comes from the fact that it not only clearly defines its trade-offs and constraints but also distinctly separates concerns, such as resource identification, resource interfaces, and definitions for interchanged data. This delineation makes it relatively easy for developers designing and building RESTful services to consider and track important issues that can profoundly impact system flexibility, scalability, and performance.

3. Proposed System

The product is entirely new product. It is not a component of a larger system. The climate web application is an online web application which displays climate data such as pressure, temperature and humidity of particular city and will interact with a RESTful web service application which provides RESTful API services in order to fetch climate data from Mongo DB. Continuous Integration system monitors the sensors data from cloud and pushes the climate data in the Mongo DB.

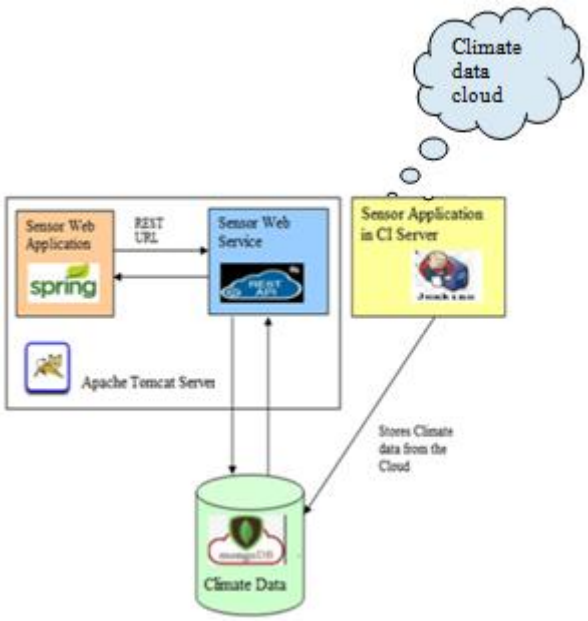

Figure 2: Architecture of Climate data application

Product Sub system:

i) **Sensor Web Application**: It is a web application that is provided in the form of URL which is a user interface to the user. User can make the selection of the city and request for weather data like temperature, pressure and humidity
ii) **Sensor Web Service**: Sensor web service is a RESTfulWeb Services, which is responsible to process, coordinate and manage sensor data stored in Mongo DB. It will exchange JSON messages to realize the data communication with sensor web application
iii) **Sensor Application**: It is a script which running continuously in Continuous Integration Jenkins server. Sensor application is an application which stores weather data fetched from cloud and pushes the data into Mongo DB.
iv) **Cloud climate data**: All the weather information like temperature, pressure, humidity are fetched from the cloud using RESTful web services are stored in Mongo Database which is a No SQL database

3.2 Product Design of Climate Data

There are three major sub components in our product which interacts with each other to realize a complete product and each components is developed in layered architecture and with existing framework which is depicted in figure below.

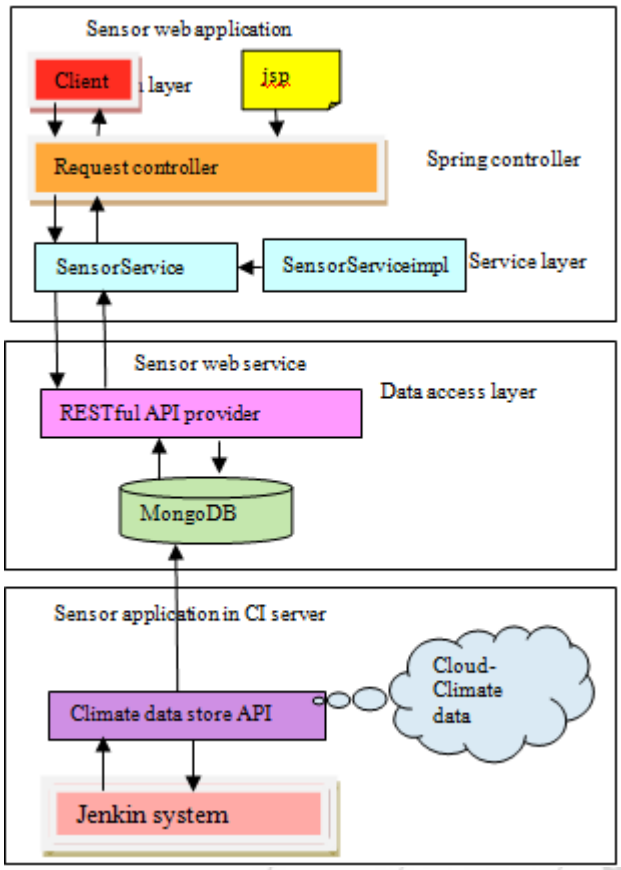

Figure 3: Detailed System design

3.2.1Sensor Web Application

It is a web application developed on Spring MVC framework which is deployed on an Apache Tomcat server. It is client application, accessed via URL and intractable application which responds to user's request by fetching data from the RESTful web server and visualize the climate data fetched from RESTful web services to the users.It provides solution to layer an application by separating three concerns presentation, controller and business flow.

- The Model in our application is Service Layers implemented in the form of SensorService which give information about request or requested information or Model can be a POJO converted from JSON which encapsulates the application data given by the controller.
- The View is web page in our application which display the information of cities and weather information and is responsible for rendering the weather data and in general it generates HTML output that the client's browser can interpret.
- The Controller in our application is RequestController and is responsible for processinguser requests for fetching weather information and building appropriate model from SensorService and passes it to the view for rendering.

Request Controller: Routes all the requests into framework control that means whenever requests land on different controllers it queues that request to the controller of framework without this MVC framework will not may be able to take control of the request at landing at the application.

In Spring MVC framework Dispatcher Servlet access Front Controller which handles all coming requests and queues for forward to the different controller.

1) Whenever request lands the dispatcher servlet consult with HandlerMapping to call the appropriate controller. (HandlerMapping- is a component which have the map of URL and Controller which need to be invoked for that particular request which lands with URL)
2) Then Dispatcher servlet has information about which controller need to be invoked
3) Then that controller will be invoked
4) And Controller can request the model for some information (about some DAO, Service layer or Data in POJO, or data in database using business logic)
5) Once process has been done then dispatcher servlet get the response then dispatcher servlet will get view resolver to build the view and view resolver look out what view has being configured it has been JSP, Velocity, XML etc. based this configuration view has been prepared and the information from model i.e. POJO it will be put on the view and response will be send back to browser.

Spring MVC Request Flow:

1) Request lands to Request Controller
2) Capture the Request Locale i.e use for internationalization i.e Read .properties files
3) Check for multipart-file(MIME type header or not) upload data from distributed application
4) Consult with HandlerMapping for which Controller to be invoked
5) And Then responsibility is given to the Handler Chain
6) This Handler Chain is responsible to be invoked some of the interceptors that needs to be invoked before of a controller and after the controller that means interceptors are here like very too similar to the filters that help to separate the pre-process logic and post-process logic.
7) After process of pre-process interceptor return to the controller process the post-process logic.
8) Then return to the view resolver prepared the view based on your configuration decide which configuration (JSP, Velocity, PDF etc.) to be invoked.
9) After choosing view technology prepare the view and return the response back to the client.

3.2.2 Sensor Web Service

RESTfulAPIProvider is a simple REST Provider-annotated component @RESTful API Provider, provides defaults for all the REST API methods to access the city information and weather data. For example, the RestAPI Provider handles all requests that start with a city name (like Bangalore)followed by /weather. Any methods in the type that further qualify the URI, like read climateData, are added to the root request mapping. Thus, read weather information is, in effect, mapped to /{cityId}/weather/{temperature}. Methods that don't specify a path just inherit the path mapped at the type level. The add method responds to the URI specified at the type level, but it only responds to HTTP requests with the verb.These REST API methods return simple POJOs – climate data (temp, humidity, pressure), etc., in all but the add case. When an HTTP request comes in that specifies an Accept header, Spring MVC loops through the configured

HttpMessageConverter until it finds one that can convert from the POJO domain model types into the content-type specified in the Accept header, if so configured. Spring Boot automatically wires up an HttpMessageConverter that can convert generic Object s to JSON, absent any more specific converter. HttpMessageConverter s work in both directions: incoming requests bodies are converted to Java objects, and Java objects are converted into HTTP response bodies.

Use curl (or your browser) to see the JSON response from http://localhost:8080/weather/bangalore Spring MVC will convert the incoming HTTP request (containing, perhaps, valid JSON) to a POJO using the appropriate HttpMessageConverter.

3.2.3 Sensor application in CI Server

Sensor application provides a REST API accessible from any networked client device for querying weather data. Climate data Map is a cloud which provides REST API. Following are the features

- Access current weather data for only location in Karnataka state of India
- Current weather is frequently updated based on global models and data from Cloud
- Data is available in JSON, XML, or HTML format

4. Results

4.1 Comparison of SOAP vs REST based on response time of request

As mentioned in the paper [7], we used the algorithm to check the response time of number of service request made from SOAP and REST to fetch the climate data. Response times for fetching all temperature, pressure, and humidity data service requests from wired clients is plotted in the below graph.

4.2 Snapshots

4.2.1 Main page displaying the details of the locations and also presenting the different icons for the user (temperature icon, pressure icon, humidity icon, complete climate icon).

4.2.2Table displaying the temperature data after climate user selecting the city and clicking on temperature icon.

4.2.3 Table displaying the pressure temperature data after climate user selecting the city and clicking on pressure icon.

4.2.4 Table displaying the humidity data after weather user selecting the city and clicking on humidity icon.

4.2.5 Table displaying the weather data after weather user selecting the city and clicking on complete climate data icon.

5. Conclusion

In this paper, we demonstrated the automation of sensor to generate the climate data information from cloud using

Jenkins Continuous Integration server which is an effective way to automate the hardware. Use of NoSQL Mongo database to store the climate data where the exchange format is based on JSON which is also exchange format with REST and better solution to handle huge amount of climate data effectively. We also designed and developed the RESTful web service application based on Spring RESTful web services where we exposed all the climate data information as a RESTful API service to the consumer. Our proposed solution achieves loose couplingand is platform independent and interoperable with many other applications with the help of layered architecture developed. We also made a comparison of REST and SOAP based web services and proved that REST is best suitable solution for Internet-scale distributed data integration. And finally, the useof a Spring MVC framework which is robust, flexible and provide a clear distinction of model, view controller and is well designedfor rapidly developing web applications.. Hence from this paper we obtained various climatic condition such as temperature, pressure, humidity etc. of various cities of Karnataka as result.

Application is now only restricted to display Karnataka major cities climate data information. It should be possible to display the climate data of major cities of all countries worldwide. Application can be extended to display the future climate data (3 to 7 days) which needs extension with GUI and storage of future climate data in Mongo DB .It can be extended to display the icons of weather indicator (sunny, cloudy, and rainy) in the GUI.

It can also include more climate data information like wind speed, precipitation information. Project can be extended to Android application to develop only the UIlayer and service can be consumed with Sensor web services.Application can also stores the past data for one month in the Mongo DB to show the effectiveness of Mongo database.

References

[1] T. Kobialka,R. Buyya, et al, ”A Sensor Web Middleware with StatefulServices for Heterogeneous Sensor Networks”, 3rd International Conferenceon Intelligent Sensors, Sensor Networks and Information, ISSNIP2007, Melbourne, Qld,491-496 (2007) .
[2] P. Amirian, A. Basiri, A. Winstanley, ”Architecture for implementing geospatial web services using SOA and NoSQL solutions”, The ThirdInternational Conference on Digital Information and CommunicationTechnology and its Applications (DICTAP2013),161-169 (2013).
[3] J. Lawler, B. Hawel. “Service Oriented Architecture:SOA strategy, Methodology and Technology”, Taylor &Francis Group (2008)
[4] Brown. A., Johnston,S., Kelly, K, ”Using Service-Oriented Architectureand Component- Based Development to Build Web Service Applications.”, A Rational Software White Paper,2002.
[5] Mahmoud. Q , ”The Road to Enterprise Application Integration (EAI)”, Service-Oriented Architecture (SOA) and Web Services publishedApril 2005, Available: http://www.oracle.com/technetwork/articles/javase/soa-142870.html, [Accessed: 30 Apr 2014].
[6] J. Meng, S. Mei, and Z. Yan, ”Restful web services: A solution fordistributed data integration”,Computational Intelligence and SoftwareEngineering, 2009. CiSE 2009.International Conference on, pages 1 4,2009.
[7] Rozlog. M, ”REST and SOAP: When Should I Use Each (or Both)”,01 Apr 2010 [Blog entry] Available:http://www.infoq.com/articles/rest-soap-when-to-use-each
[8] B. Upadhyaya, Ying Zou, Hua Xiao, J. Ng, and A. Lau, ” Migration ofSOAP-based Services to RESTful Services”,Proc. 13th IEEE InternationalSymposium on Web Systems Evolution (WSE), 2011, pp. 105114.
[9] H. He, Implementing REST Web Services: Best Practices and Guidelines.11 AUG 2004.[Blog entry], O’REILLY XML.com. Available http://www.xml.com/pub/a/2004/08/11/rest.html
[10] “RESTful Web Services Development Checklist” Internet Computing,IEEE Volume: 12 , Issue: 6. Nov.-Dec. 2008.
[11] Apache WSRF: http://ws.apache.org/wsrf/
[12] Liang S, Coritoru, A., Tao, C. A Distributed Geo-Spatial Infrastructure for Smart Sensor Webs, Journal of Computers and Geosciences Vol.31, 2005.
[13] XML Beans: http://xmlbeans.apache.org/overview.html
[14] A. Alowisheq, D. E. Millard, T. Tiropanis, “ EXPRESS: EXPressingREstful Semantic Services Using Domain Ontologies.” International Semantic Web Conference 2009: 941-948
[15] A. K. Jain, M. N. Murty, P. J. Flynn; Data clustering: a review; ACM Computing Surveys (CSUR) Volume 31 Issue 3, Sept. 1999