REST API: An advance technique to develop light weight application

Shiv Kumar Goyal1
Deputy HOD of Master of Computer Application
Vivekananda Educational Society's Institute Of Technology,
Chembur, Mumbai, Maharashtra
shivkumar.goel@ves.ac.in

Prashant Kashinath Chaudhari2
Deputy HOD of Master of Computer Application
Vivekananda Educational Society's Institute Of Technology,
Chembur, Mumbai, Maharashtra
prashant.chaudhari@ves.ac.in

Krunalsinh Gajrajsinh Vala3
Vivekananda Educational Society's Institute Of Technology,
Chembur, Mumbai, Maharashtra
krunalsinh.vala@ves.ac.in

Adwait Vijay Kulkarni4
Vivekananda Educational Society's Institute Of Technology,
Chembur, Mumbai, Maharashtra
adwait.kulkarni@ves.ac.in

***Abstract*—**Due to increasing number of user over the internet it has become apparent to find suitable replacement for the old technologies such as SOAP and RPC as they cause network latency. The REST is designed to improve performance by increasing scalability, generality of interfaces, independent deployment and allowing intermediary components.

***Keywords*—** *REST API, Bot.*

*****

I. INTRODUCTION

In order to develop a web based application to meet today's standards the app has to be scalable and performance should not include any latency issues. The web has scaled from few requests a day to few thousand requests per minute which puts and enormous pressure on server recourses .Before the REST was introduced Industries used Remote Procedure Calls (RPC) to share and receive data among web applications spread over the internet but using RPC like standards for message passing put severe limitation in terms of performance and scalability. In modern web application where new features are introduced by Industries every day we cannot reply on old approach of big bang deployment where all services are deployed at once.

We have structured the article as follows:
1. Establishing understanding about REST
2. Application Developed using REST

II. ABOUT REST

Roy T. Fielding defines REST in a coordinated set of architectural constraints that attempts to minimize latency and network communication while at the same time maximizing the independence and scalability of component implementations.

REST basically works on coordinated set of data elements, connectors and components where hypermedia system is spread across multiple systems and the main focus is on interaction between elements and roles by each of the components rather than how the actual implementation takes place.

III. UNDERSTANDING REST

REST API is an architectural style of World Wide Web. It increases application performance, scalability, reliability and modifiable.

REST specifies six constraints:

A. Uniform Interface

REST acts as an interface between server and client's application. Instead of sending complete database to client, REST sends some part of data in JSON, XML, and HTML or in plain text. With proper permissions client can request data, modify or delete data from resource server. Client can request REST by query-string parameters, in body of request, in header and requested URI.

B. Stateless

REST maintains state with client's application using necessary parameters within the request itself. It can be a part of requested body, query-string or URI. When all data transaction is done, the current state is sent back to client via headers or request body.

C. Cache

Client can cache response given by REST on WWW. REST helps to make responses cache-able implicitly or explicitly. Once the response is cached, it can be reused for further request which increases performance and reduces network latency.

D. Client-Server

REST helps developers to implement MVC architecture where REST acts as a Controller, client's application acts as View and the business logic acts as Model. Because of this, client and servers interface are separated uniformly. As servers are not dependent on user interface they can be easily upgraded or expanded.

E. Layered System

REST hides server side architecture from client. That makes gives the server capability to improve by load-balancing and providing cache. Client application never

knows whether it is making request to end server or intermediary server.

F. Code-On-Demand

Client can extend its application functionality by downloading and executing code in the form of scripts. Due to this system extensibility increases but at the same time visibility of code decreases. Because of this reason Code-on-demand is the only constraint that is optional.

IV. APPLICATIONS OF REST API

REST is somewhat similar to Web Services and RPC. Though it looks simple, REST is fully-featured; there's basically nothing you can do in Web Services that can't be done using REST.
Like Web Services, a REST service is

Platform-independent: It doesn't matter if the server is UNIX based; the client is using Windows, or anything else. REST services are independent of the type of operating system both client-servers are using.

Language-independent: From a developer's point of view it doesn't matter in which programming language you create REST Services. An application which is developed and based on Java can easily access, communicate and consume REST services created using C#.

Standards-based: It runs on top of HTTP. For security, username/password tokens are often used and in case of encryption, REST can be used on top of HTTPS (secure sockets). It can easily be used in the presence of firewalls.

The main advantage of REST over Web Services is the ease of implementation, agility of the design, and the lightweight approach to things.

REST is loosely coupled. We can easily update the server without having to update the clients. If a piece of software is updated on server, the client can easily access the updated services, without any need to update its machine.

REST is used when somebody needs something up-and-running quickly, with good performance and low overhead. With better cache support, lightweight requests and responses, and easier response parsing, REST reduces network traffic, too.

Many service providers use REST API. In addition to REST, some of them also support a WSDL (Web Services) API. It's up to client to pick which one to use, but in most cases when both alternatives are available, REST calls are easier to create, the results are easier to parse and use, and it's also less resource-heavy on client system.

Google APIs: Google APIs support most modern sites online. From analytics, to the advertising exchange, maps, to Google+ shares, chances are, most sites are using a Google API. Currently, 57 APIs are available from Google like Cloud APIs, YouTube APIs, Advertising APIs, Maps APIs, Translate APIs, Custom Search APIs, etc.

Facebook API: Facebook APIs enables developers to integrate a more personalized social experience on their own websites, including the ability to "Like" and share pages. Through the Facebook API, Facebook data can connect with countless business apps for endless possibilities.

Twitter API: Twitter is a real-time information network that continues to allow millions of people to connect and share information for free. With the Twitter API, programmers have the ability to integrate with everyday personal and business applications, creating automation and quick accessibility for sending and reading tweets.

Amazon API: Both Amazon S3 and Amazon EC2 web services offer great flexibility. With S3, we can store our files in Amazon's data center and manage access using the Amazon API, which has both a REST and SOAP interface.

Word Press API: Word Press API allows many applications to connect with its system via installable Word Press plug-in, extending site functionality to incorporate social networks, play multimedia, and much more. Using the open source Word Press API, companies are able apply new functionality and features to Word Press while instantaneously leveraging a new sales channel.

Flicker API: The Flicker API allows for development of different ways to share and organize photos, including integration with other social apps. Flicker also encourages programmers to converse, share, and curate using the Flicker API.

As REST matures, we can expect it to become better understood and more popular even in more conservative industries.

V. IMPLEMENTATION OF REST API

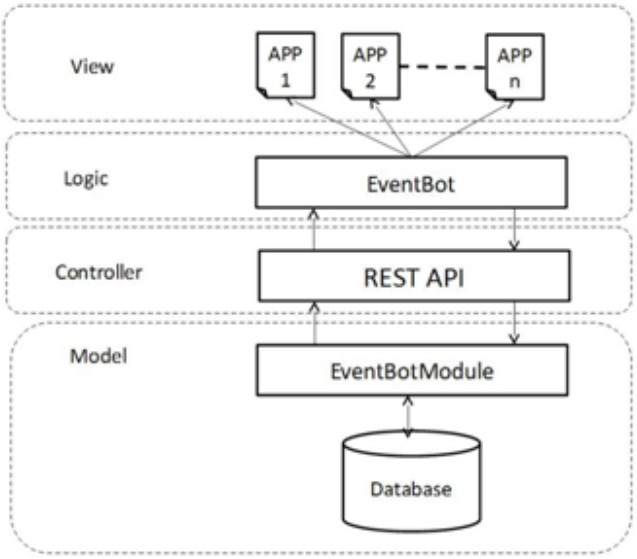
A bot is a computer program which makes conversation by text with user. But with the increase in chatting applications, number of users has also increased. To deal with such huge audience we need REST API.

Any organization or industry faces many issues while managing an event. To solve problems there are many mobile applications available for event organization. But with increasing necessity of mobile cloud computing there is demand of REST API based app development.

This help to develop application that is light weight in terms of data communication over network. It improves

application performance and efficiency by reducing time-consuming tasks.

A. Using the concept of REST API, we have developed



EventBot using REST API

EventBot help users to manage an event very easily. REST API methods used by EventBot

POST: To create event, to get event based identity number using AUTH token and to post event queries we make a POST request to REST.

GET: To get event information we make a GET request to REST which returns complete information of an event based on event id.

PUT: PUT request is made to update entries of participants for a particular event.

DELETE: When a participant leaves an event, his/her entry is deleted from that event.

B. Response of REST

EventBotModule processes all the requests that comes through REST API and generate a JSON response which is sent to the client application. This reduces client - side processing as client only makes HTTP requests and gets response for the same.

Based on the HTTP response codes, client can decide whether to process the JSON response or not.

C. HTTP response codes that a client application gets for a request

ACKNOWLEDGMENT

We take much pride in presenting our paper. In our path, we definitely need to mention the names of certain individuals without whose assistance our project would have been a difficult undertaking indeed.

We are pleased to have this opportunity to express our deepest gratitude to Prof. NISHI TIKU (Head of Department) and Prof. SHIVKUMAR GOEL whose valuable guidance and suggestions helped us in accomplishing our project.

REFERENCES

[1] http://www.restapitutorial.com – RestAPI Tutorial
[2] https://en.wikipedia.org – Information about REST
[3] http://www.ics.uci.edu/ - REST documentation