JSON Based Web Services Virtualization Platform

K.D.I.U.K. Gunawardana

2019

JSON Based Web Services Virtualization Platform

A dissertation submitted for the Degree of Master of Computer Science

K.D.I.U.K Gunawardana
University of Colombo School of Computing
2019

Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name: K. D. I. U. K. Gunawardana

Registration Number: 2016/MCS/035

Index Number: 16440351

-- ----------------------------------

Signature: Date:

This is to certify that this thesis is based on the work of

Mr. K. D. I. U. K. Gunawardana

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name: Prof K. P. Hewagamage

-- ----------------------------------

Signature: Date:

Abstract

Delivering working software on time is a major challenge in the software industry. This challenge becomes more challengeable when the initial requirements want to be changed while the development is in progress. During the last few decades, different software development methodologies have been emerged by different groups to minimize those problems and improve the quality of the development process, hence, to improve the quality of the final product. Currently, the dimension of this problem has become more complex since the emergence of distributed software systems. Nowadays, most of the enterprise level applications are heterogeneous, consists of different service, application modules and communication patterns. And Web services are considered as the most efficient communication method among applications. In such an environment, there should be a well-organized engineering process to produce quality output. The parallel development process is considered as one of the best practice to improve efficiency by sharing the responsibilities of different components to different engineering teams through documentation and meetings. Though, communication of requirements changes is harder and takes more time. Sometime it may cause another team to hold their works until another team completes their tasks. This is one of the major drawbacks of the parallel development process. This dissertation proposes a solution to overcome this problem and improve the efficiency of the parallel development process through virtualizing web services using JSON. The proposed solution contains two major parts. First part handles virtualization of different services and the second part handles arbitrary data generation while processing the HTTP requests. The proposed solution uses JSON as the key format of persisting service definitions since it's ease of maintainability and accessibility over XML and traditional database schemas. Data generation part can process requests and produce complying formatted random data asynchronously for each request. Simple user interfaces allow users to work with the system without deeper knowledge about computer science. So technical and non-technical people like business analysts can interact with the system and communicate requirement changes rapidly to other stakeholders.

Acknowledgements

This research would not have been successful without the help, guidance and dedication of several persons who actually contributed their valuable time on my effort.

First and foremost, I would like to express my project supervisor Prof. K. P. Hewagamage who has supported me through the project. His guidance and push through the whole year lead me to complete this project successfully. Without his kind contribution, I could not have achieved my objectives.

Also, I would like to thank the MCS coordinators, project coordinators, and academic staff and the library staff who help me in numerous ways to complete this project successfully.

I wish to give my special thanks to my mother, my father and my loving wife who gave me the courage to overcome every difficulty during this period and finish this project successfully.

I would like to give special thanks to all my friends who gave me courage and help to finish this project successfully.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

HTML - Hyper Text Markup Language

HTTP - HyperText Transfer Protocol

IDE - Integrated Development Environment

JSON - JavaScript Object Notation

RDBMS - Relational Database Management System

SVN - Subversions

TS – TypeScript

UI - User Interface

UX - User Experience

XML - Extensible Markup Language

Chapter 1

Introduction

Since the invention of the computer, the complexity of computer-based application has been changed drastically. Now the variations of software are changed from a simple console based application to huge and complex enterprise application which runs on different platform varies from a single stand computer to multiple supercomputer or virtual environments. These systems contain million number of lines of the codebase. Additionally, they have consumed a huge amount of man-hours workload to build up a fully functional software system. Even though, most of those systems contain at least a few bugs. So the process of software development field has also expanded through many disciplines to handle the building of complex systems. It is still being developed lots of processes to improve software development processes to achieve a goal of bug-free software within reasonable man-hour consumption.

Modularizing a large system into a number of small subsystems based on the core functionality of each system is considered as a best practice in the field since the ability to manage the complexity of the project easily. Service-oriented architecture(SOA), microservices and layered architecture are some of those approaches to modularize large systems into small subsystems. Modularizing approach also helps to develop and test each component independently.

Communication among components can be achieved a few techniques such as remote method invocations and web services etc. Web services are the most commonly used mechanism.

1.1 Motivation

One of the main goals of software development processes is delivering a defect or bugs free product on time and within the budget. This task is very hard for any software development team, even for very skilled and experienced teams. Today, there are many development processes and practices to be developed to achieve this consistent goal.

A process of bug-free software development is a combination of developing and testing practices. Having good testing skills is so important as well as having a skilled development team who can perform best software development practices. Talented testing teams can help to deliver quality software through proper tastings.

Good software architectures can combine those aspects through their architecting skills. SOA and Microservices are two of the most popular architectural patterns these days since the ability to divide large software systems into smaller parts which are more manageable. Agile and scrum are also very popular methodologies since the ability of handling requirement changes more efficiently and reliably than ever before. Therefore, a combination of SOA, Microservices, Agile and Scrum can be seen most of the time in the industry.

1.2 Problem

Today's software solutions are very complex and consist of a variety of applications and service modules which are mainly interacted with each other through web services. Since the serial development takes too much development time, the Parallel development process is considered as the most efficient ways to implement these type of solutions. Parallel development processes allow different development teams to collaborate so each development team can work on separate modules of the entire project. These teams may or may not be working on the same technology. Some teams may be working on the backend, some teams may be working on middleware and some teams may be on the user experience side. Middleware which is mostly web services will combine all together to build up the entire system.

In the test phase, standalone tests are done at first level to identify unit level issues in systems then integration testing is performed to check whether how it works with other systems. After that, the entire system will be tested to deliver a quality system. Testing and fixing phases will be done iteratively until it finds a bug free product.

But the problem is this parallel implementation of the subsystems is directly depended on the implementation of the web service. Even in a place where documentations are properly maintained, it is hard to implement and deliver changes quickly. Traditional development processes are keen to documentation but new development processes such as Agile never intend to keep such a bunch of documents, it always tried to keep as less as possible.

Change requirements are also a compulsory thing in any software product. A change request is a proposal to alter a product or system, often brought up by the client or another team member. During a project, this can happen when a client wants to change or alter the agreed upon deliverables. Change requests can also be initiated internally as well and can include things like changing or upgrading software. In general, there are two types of change requests: those that are inside the scope and those that are outside the scope of the project. Change requests that are inside the scope involve small corrections to an existing requirement. On the other hand, change requests that are outside the scope take a considerable amount of time to implement and have a more sizeable impact on the budget. A change request is often inevitable and should be expected at some point in any project. When the entire team is up-to-date on the change request it can be dealt with in an appropriate and timely manner. It is the change requests that are not approved or not communicated to the other team members that ultimately cause a problem. Once a change request has been made, the entire team should be informed and they can come to an agreement about how to satisfy the request without using unnecessary resources.

The main purpose of this project is to address this issue and define a way to implement web services virtually. So developers can continue development works with virtual services until the real services are released.

1.3 Aim

Providing a seamless integration of web services with depending systems is the main aim of this project. Seamless integration is supposed to achieve through a virtualization mechanism and an arbitrary data generation process. Virtualizing mechanism allows other systems to feel that the requested resource is available. Then, Data generation part can provide proper data for virtual services.

1.4 Objectives

Objective 01 – Implementing a simple data format to store and to retrieve service definition data for fast execution.

Objective 02 – Implementing a mechanism to process HTTP request data and to identify the service part of the request to reduce the latency.

Objective 03 – Implementing a mechanism to generate variety of arbitrary data efficiently.

1.5 Scope

The proposed solution will provide a solution to virtualize HTTP Rest web services. So the users will be provided with a user interface to describe their services simply as the main function. Additionally, they may be received facilities to manage services, users, and teams of the project. In the service definition part, it will provide some basic support of the basic mathematical functions such as plus, minus, less than/greater than etc. to describe some constraints of the data. So users will be offered privileges to access multiple projects simultaneously. From the backend side, the system may generate arbitrary data for services based on user-defined data types and will return them in JSON format. Request/response log which is exposed via the interface will allow users to debug applications.

1.6 Contribution

This project contains a few challenges that should be studied further. One of the major things is keeping the service description data inside the system converting them into an intermediate state. Since there are few subsystems that use these data, it should be kept in a simple and common format for all systems. Another thing that it needs to be studied further is how to simulate behaviors of a multipart request. Finally, the arbitrary data generation part should be studied further to provide meaningful data to requests.

1.7 Structure of the thesis

The second chapter is the literature review which will discuss the ongoing works, projects, and researches on the topic. Chapter 3 will discuss about the design of the proposed solution. In the chapter 4, it will discuss about the actual implementation of the solution in detail.

Chapter 2

Literature Review

2.1 Introduction

The problem discussed in the previous chapter is not a newly found issue in the software development field. It was there from the beginning and managers could handle it by their own experience. But the necessity to solve this problem is still significant than ever since the complexity of solutions is getting complex and multiple technologies are involved to solve a problem. This chapter will discuss an analytical overview of the significant literature published on this topic.

2.2 Factors for Success and Failure of Projects

According to the CHAOS report [1] which is published by The Standish Group (https://www.standishgroup.com/) in 1995, the US spends more than $250 billion each year on IT application development of approximately 175,000 projects. The average cost of development for a large company is estimated at $2,322,000 while this becomes $1,331,000 and $434,000 respectively for medium and small range companies [1]. It also says that 31.1% of projects are canceled before they start and 52.7% project is running over the estimated budget and that value is estimated as 189% of the original budget [1].

In 1995, The Standish Group surveyed IT executive managers of different companies to identify why projects succeed while some are failed. The resulting report shows a number of factors that effect on projects as shown in the table 1 below. User involvement, executive management support, a clear statement of requirements and proper planning are considered as top priorities [2].

Project Success Factors	**% of Responses**
User Involvement	15.9%
Executive Management Support	13.9%
Clear Statement of Requirements	13.0%
Proper Planning	9.6%
Realistic Expectations	8.2%
Smaller Project Milestones	7.7%
Competent Staff	7.2%
Ownership	5.3%

Clear Vision and Objectives	2.9%
Hard-Working, Focused Staff	2.4%
Other	13.9%

Table 2-1Project Success Factors

On the other hand, that report could identify a number of factors which can be considered as the challenging factors of a project. Top priority factors are Lack of User Input, Incomplete Requirements and Specifications, Changing Requirements and Specifications. These three factors nearly contribute to 37% [2] success of the project.

Project Challenged Factors	**% of Responses**
Lack of User Input	12.8%
Incomplete Requirements and Specifications	12.3%
Changing Requirements and Specifications	11.8%
Lack of Executive Support	7.5%
Technology Incompetence	7.0%
Lack of Resources	6.4%
Unrealistic Expectations	5.9%
Unclear Objectives	5.3%
Unrealistic Time Frames	4.3%
New Technology	3.7%
Other	23.0%

Table 2-2 Project Challenge Factors

2.3 Software Engineering Practices

Well-disciplined software engineering practices can improve the quality of products and reduce the number of defects. Skills and knowledge of engineers are very important to complete a project successfully. Therefore, having highly skilled engineers is better than mid-range engineers [3].

The varying engineering team cannot guarantee the quality of the product. Manages must identify the required number of engineers as well as the required skill set before starting the project. Involvement time of different skills into the project may be different based on the phase of the project. In some cases, managers try to put more engineers into the project at the last time to finish works quickly. Adding more engineers to a late project doesn't accelerate its completion, instead of that, it takes more time since new engineers want to study the system [3].

Communication mechanisms of modern application components are heterogeneous. Different protocols and message formats are involved. So there is a wide variety of channels for users to collaborate with enterprise applications. Each channel is configured to incorporate with other systems to provide sufficient services efficiently. Therefore, Integration of enterprise applications has become a risky and important task. Integration testing is also very important to control the data integrity, just like "Even with an army of women, it still takes nine months to make a baby" [4]. Another paper has identified the enterprise integration as “the right information at the right place at the right time for decision-making” [5] through the integration of horizontal and vertical approaches. Horizontal approach is described as the integration of business-to-business. Then, the vertical approach is business-to-manufacturing integration.

2.4 Parallel Development

Parallel development process can be implemented within a standalone application development environment as well as complex enterprise level applications. In standalone projects, Parallel development can occur at the level of a single file or other configuration items (micro level) or at the level of an overall project configuration (macro level) [6].

A parallel development is required when there is a need of separate development path [6], so that there is no longer a single "latest and greatest" version, but instead two or more concurrent "latest" configurations. The necessity of parallel development of a standalone application includes below requirements [6].

1. Release preparation
2. Post-release maintenance (segregated from new development)
3. Tailored or customer-specific software
4. Segregation of work by different development teams or individuals
5. Segregation of work on different features
6. Deployment of different software variants into different environments

2.5 JSON vs XML Processing

Software components can communicate with each other using JSON or XML. In SOA, this is used very regularly. Each of these formats has its own pros and cons when parsing and processing.

1. Performances of processing JSON and XML can be compared using the below parameters [7].
2. Marshalling time: This is the time taken, in nanoseconds, by the JVM to convert an object into a stream of data
3. Unmarshalling time: This is the time taken, in nanoseconds, by the JVM to construct an object from a given stream of data.
4. Stream size: The size of a marshalled object in Bytes.
5. Memory footprint: The total memory in Bytes used during runtime from the JVM heap.

Marshaling and Unmarshalling performances of both formats can be considered as a key measurement for a selection. Jackson [9] API for JSON and JAXB API (JSR 222)[8] for XML can be used to implement marshaling and unmarshalling logic.

Marshaling performance of XML and JSON can be illustrated as below [6].

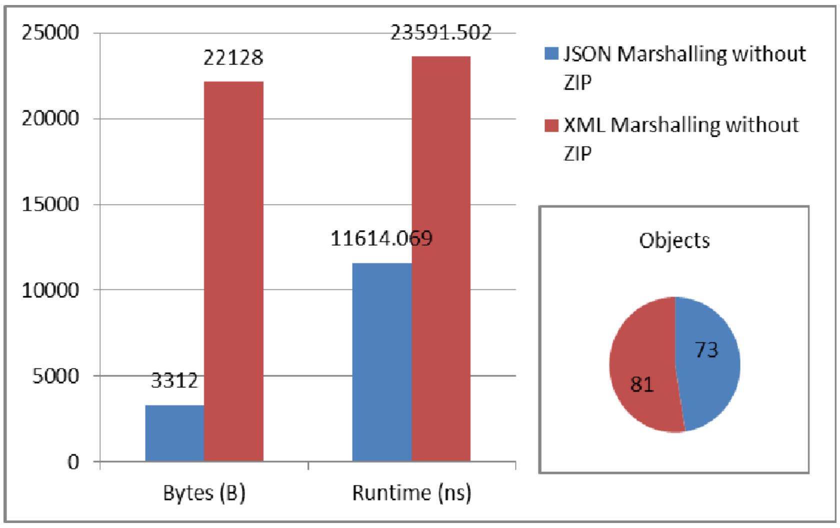


Figure 2-1 Marshaling Performance Comparison

Unmarshalling [7] performances of XML and JSON.

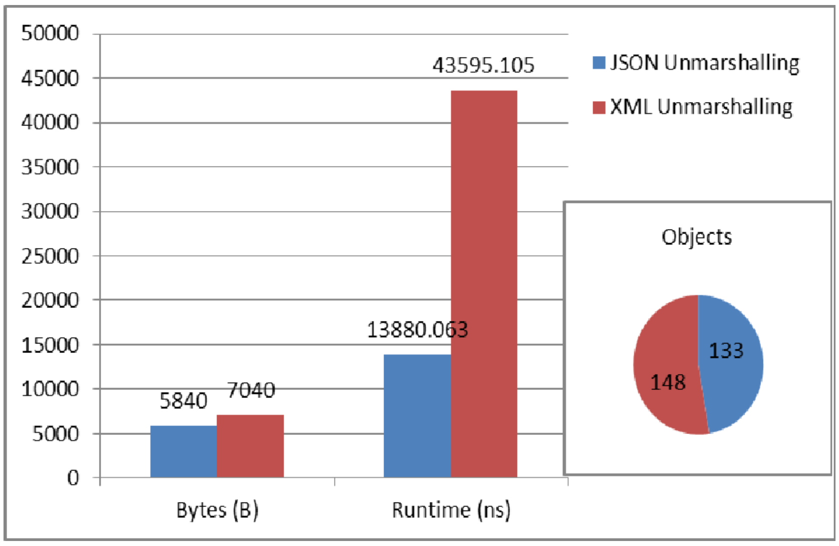


Figure 2-2 Unmarshalling Performance Comparison

As per the above benchmark data, the below conclusions can be derived [7].

1. When performance is a design consideration then JSON is better than XML.
2. When memory footprint of an application is needed to keep as small as possible then JSON is better than XML.
3. When data sharing through the internet is needed to be optimized then JSON is better than XML.
4. XML is preferred when there needs to be strict adherence to a standard protocol for communication.
5. When there is a possibility of changing the format of data sharing, then JSON is better than XML since change are taken less time.

2.6 Tools Available in the Market

2.6.1 Traffic Parrot

Both testers and developers can use this tool to do service virtualization, mocking, and simulation [10]. It supports different protocols such as HTTP(S), JMS, IBM MQ and File transfer protocols. It also supports containerization technologies such as Docker, Kubernetes, and OpenShift. This tool is ideal for testing micro services.

2.6.2 ServiceV Pro

Developers and testers can create, configure, and deploy virtual APIs with ease- increasing system availability and reducing QA costs and bottlenecks [11]. Users can easily create virtual APIs via recording live traffic, building out a RESTful service in our UI, or simply importing a Swagger or WSDL spec, then share the service with other team members.

2.6.3 Wiremock

HTTP based API simulation tool [12]. It allows checking of an edge case and failure modes that the real API may not able to produce. Additionally, it has robust and power API URL request matching, Record and Playback and Hosted Mock API service features.

2.6.4 Mountebank

NodeJS based cross-platform, multi-protocol testing tool [13]. It supports SMTP, HTTP, TCP, and HTTPS protocols. it also provides service virtualization service free of cost without any platform constraints. This tool is considered as updated, mature, and stable tool.

2.6.5 Hoverfly cloud

Hoverfly cloud is an integrated service virtualization solution [14] which is designed for integration, automation, and performance. It's deployable on Google, AWS, Google & Azure cloud platforms. Additionally, it provides facilities for reporting and on demand performance measurements.

2.6.6 MicroFocus Data Simulation Software

MicroFocus Data simulation software to virtualize micro service's behavior [15]. The tool helps to create a simulation of application behavior Allows modifying data, network, and performance models. Service Virtualization features integrated with Performance Center, ALM, LoadRunner, and Unified Functional Testing.

2.6.7 Parasoft Virtualize

Parasoft Virtualize helps to create a realistic simulation of test dependencies [16]. It allows easy configuration of complex test conditions as well as separating testing process from time-boxed access to external system. Also, this tool can be used to find hidden performance issues in the application under the test.

2.6.8 CA service Virtualization

CA Service Virtualization tool which supports more parallel developments simulates unavailable systems across the software development lifecycle [17]. It simplifies the management of development and testing processes. And helps to streamline development by virtualizing dependent systems including mainframes, and external service providers.

2.6.9 Mocklab

Mocklab is service virtualization tool which allows easy copy, paste or record stubbed HTTP responses [18]. It can virtualize test edge case and failure modes which the real API never able to produce.

2.6.10 Rational Test Virtualization Server

IBM Rational Test Virtualization offers fast and quick testing of services, software, and applications [19]. It helps to reduce dependencies by simulating part or an entire application. Additionally, it supports for sharing and reusing virtualized environments, integrating with other tools and supporting middleware etc.

2.6.11 Tricentis Tosca

Tricentis Tosca is a tool to test highly interconnected systems with many components evolving in parallel [20]. Also it can use to simulate Interactions necessary for Testing.

2.7 Chapter Summary

Chapter 2 has discussed the studies regarding service virtualization and some popular service virtualization tools currently available in the market. Additionally, it discussed software engineering practices and JSON/XML data sharing formats. Then, the basic features of the currently popular service virtualization tools.

Chapter 3

Analysis and Design

3.1 Introduction

This chapter elaborates the theoretical background of the procedures that is followed in order to satisfy the objective of this project. Further, important aspects used for implementation purposes are discussed.

3.2 Overview

The proposed solution to improve the efficiency of parallel development processes is based on the virtualizing web services. Main requirement of virtualizing service is to decline the dependencies of other modules with web services. Virtualization allows developers to continue development the development process as there are required web services. Later, these virtual services will be replaced from actual services in the integration steps.

The solution contains facilities to declare services and to generate data. Service declaring part provides required methods and interfaces to define web services, data types and their data generation logic. Then, data generation part does the generating of arbitrary data based on user defined restrictions in the service declaring part.

Figure 3.1 shows the high level architecture of the solution. Then, it shows two major parts in dashed lines and its sub components separately. Additionally, the arrows indicate the communication path and directions.

Rest of this chapter will be discussing about the service declaring and data generating parts in detail.

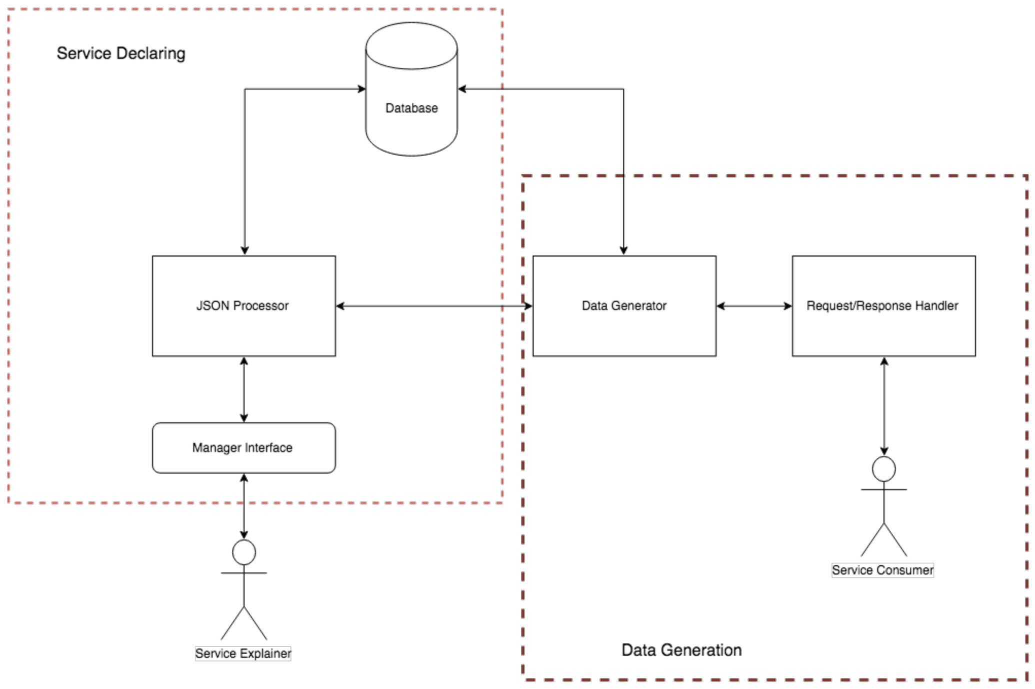


Figure 3-1 Architecture of the solution

3.3 Service Declaring Module

Service manager is responsible for facilitating users to add service information which should be virtualized and save those data as an intermediate format in the persistence layer as data or JSON files. The service layer contains three main parts,

1. Manager Interfaces
2. JSON Processor
3. Data Persistence Layer

3.3.1 Manager Interfaces

This part contains user interfaces which are provided to users for declaring web services. Interfaces will be provided for below tasks.

1. Crate a new web service
2. Manage existing web service
3. Publish and unpublish web services
4. Manage user authorizations for web services

3.3.2 JSON Processor

JSON processor can convert service definition into a JSON and read data from a JSON. JSON conversion part is important to the service declaring module and JSON reading part is important to data generation part. So data generation process of the JSON processor will be discussed under the topic 3.3.

Data conversion part maintain a default JSON format for all services. When new service going to be declared then it is formatted based on the default JSON format. Unique JSON format for all services will make the processing part easy in the data generation module.

3.3.3 Data Persistence Layer

Persistence layer store the data of the system. This data may be user related or service related. User related data will be stored in a relational database. Service related data is kept as JSON so this can be store in the relational database or in the normal file system of the server.

3.4 Data Generation Module

Generating precise arbitrary data for user request based on its constraints is the main task of this module. There are two sub modules for this module one is to do the actual data generation and other one is to handle user requests.

Figure 3.2 shows the basic structure of the data generation module. Whole module has designed as a set of layer since it helps to improve the maintainability of the entire module. There are some parts which shares the component with the service declaration part such as JSON Processor. Other parts are unique to the data generation module.

Basic layers of this module can be listed as below.

1. Request/Response handler
2. Manager layer
3. JSON Processor
4. Persistence layer (Database)
5. Utility layer

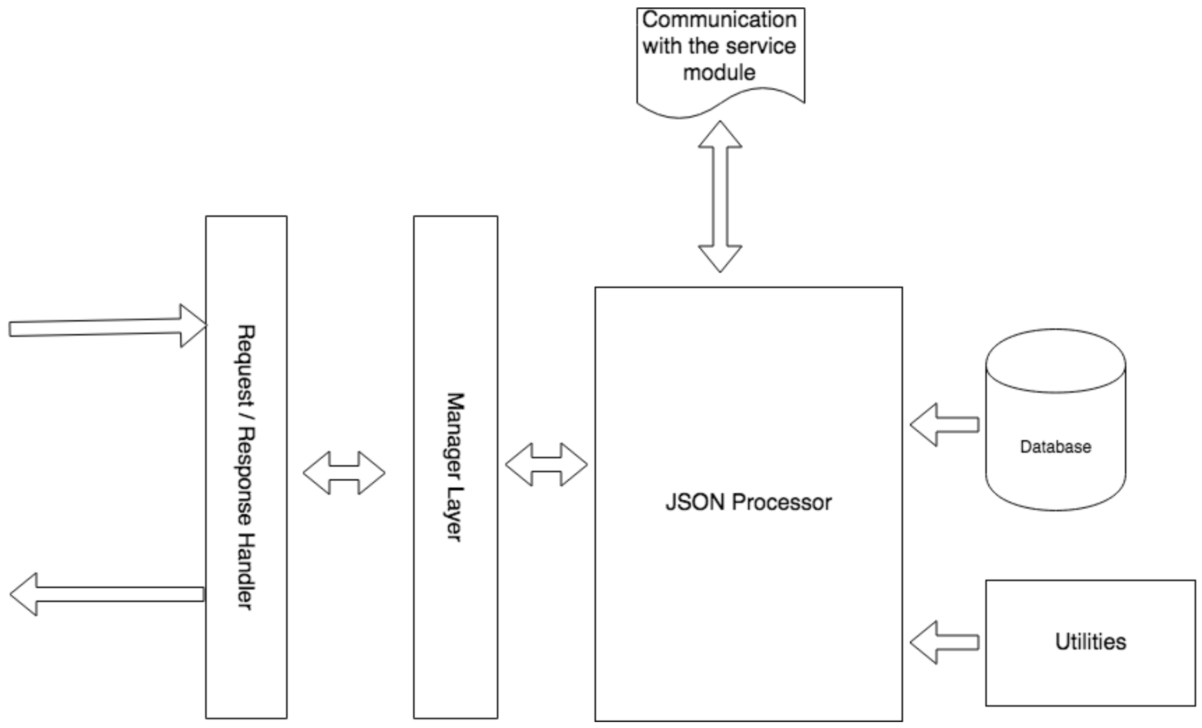

Communication with the service module
Request / Response Handler
Manager Layer
JSON Processor
Database
Utilities

Figure 3-2 Data generation module

3.4.1 Request-Response Handler

In the real environment, the system may receive large number of request from different users. So there should be a mechanism to handle request and responses properly. Otherwise, the system may fail. It may produce wrong data. And also, it will take more time to response for a request.

Request-response handler maintains a queue to hold requests. Then it will process each request one by one in FIFO manner. The system contains few threads to increase the performance of the processing part.

3.4.2 Manager Layer

Manager layer behaves as a mediator for request and data. It takes requests from the request-response handler and analyze it, then calls to proper methods in the JSON processor to bind the data with the response. Finally, it passes response data to the request-response handler.

3.4.3 Persistence Layer

See topic 3.2.3

3.4.4 JSON Processor

See topic 3.2.2

3.4.5 Utility Layer

Utility layer contains number of utility classes which are required to generate data such as sample string generators, data formatters, random number generators etc.

3.5 User Management

User management part handles basic operations on users and services. Main task of this module can be listed as below.

1. Adding user to the system
2. Editing users of the system
3. Deleting user of the system

This module also handles how to manage users within groups and services. Group is a set of users such as a project team. Individual users or groups can be assigned to a single service or bunch of services. Only the authorized users or groups can access the service through a given URL.

Basic operations related to this part can be listed as below.

1. Creating groups
2. Edit/Delete groups
3. Assigning users to a group
4. Assigning users or groups to a service

3.6 Chapter Summary

In this chapter, it is discussed about the design part of the proposed solution for web service virtualization. Basic structure of the solution is saving service information as JSON and generating data based on those JSON.

Relational data access consumes more resources and time to access databases by falling the performance of the system down. If the system is going to access the database more frequently then this effect is high. So the solution proposed to use JSON as the service definition which allows to access only once to get service data and the process that JSON to generate sample data for requests.

There will be two type of interfaces for users to define and consume services. Flat UIs will be provided to manager services and users. Access point will be provided to consume services.

Chapter 4

Implementation

4.1 Introduction

This chapter will discuss about the implementation of the solution that discussed in the Chapter 3 by providing a clear idea about how each module performs its functions and why those particular modules are there in the system.

This chapter contains may topics to explain the solution implementation. Topic 4.2 will discuss about the technology stack that the proposed solution is going to use. Then in the topic 4.3 will explain about the implementation of the service declaring module in detail. Finally, the topic 4.4 will explain about the implementation of the data generation module. Additionally, each topic will include subtopics for further explanation of some complex implementations.

4.2 Implementation Environment and Technologies

The proposed solution is intended to run on a web server. Latest WildFly application server will be used for the hosting purpose. Front end will be developed using AngularJS. Then Java will be used for core system development. MySQL will be used as the persistence layer.

WildFly is an application server developed by Red Hat and it is more robust and open source application server. It provides variety of JEE services which can be used to implement the solution. Additionally, there are number of third party application which support the WildFly server.

AngularJS is the latest front end development framework developed by Google. It provides facilities to implement systems as a collection of components with a single HTML page. Component based approach is very manageable and easy development than traditional multiple page applications.

Java is the most trusted application development language with huge number of third party libraries and frameworks for variety of purposes. It supports object oriented development techniques so developers can implement the systems as a collection of objects.

MySQL is one of the leading opens source data persistency tools that supports RDBMS. This tool is provided by Oracle corporation which is also an industry leader in the software field. Also, it is come up with bunch of other supporting tools such as MySQL Workbench which can be used to development of database as well as the maintenance of the database. Then it contains many Java connectors so developers can easily develop software to work with this server.

Development environment will be different for Java and AngularJS. JetBrain's IntelliJ IDEA is a one of the most popular IDE for Java development. This tool provides number of facilities to enable developers task easy such as auto suggestion, smart completion of codes, code review facilities and SVN integrations etc.

For Angular related developments, Mircosoft's Visual Studio Code will be used for TypeScript related implementations since it includes support for debugging, embeddedGit control, syntax highlighting, intelligent code completion, snippets, and code refactoring etc.

MySQL workbench is an open source and free software for designing databases for MySQL servers. This provides more graphical views to design databases such as interactive schema diagrams. So developers can see entire database at once with their relations and other constrain parameters.

4.3 Service Declaring Module Implementation

There are three major development layers under the service declaring module. Let's discuss about each layer separately. In the interim report those, these items will not be discussed more thoroughly since the development phase is still underway and any change is possible.

1. Front-end layer
2. JSON Processor layer
3. Persistence layer

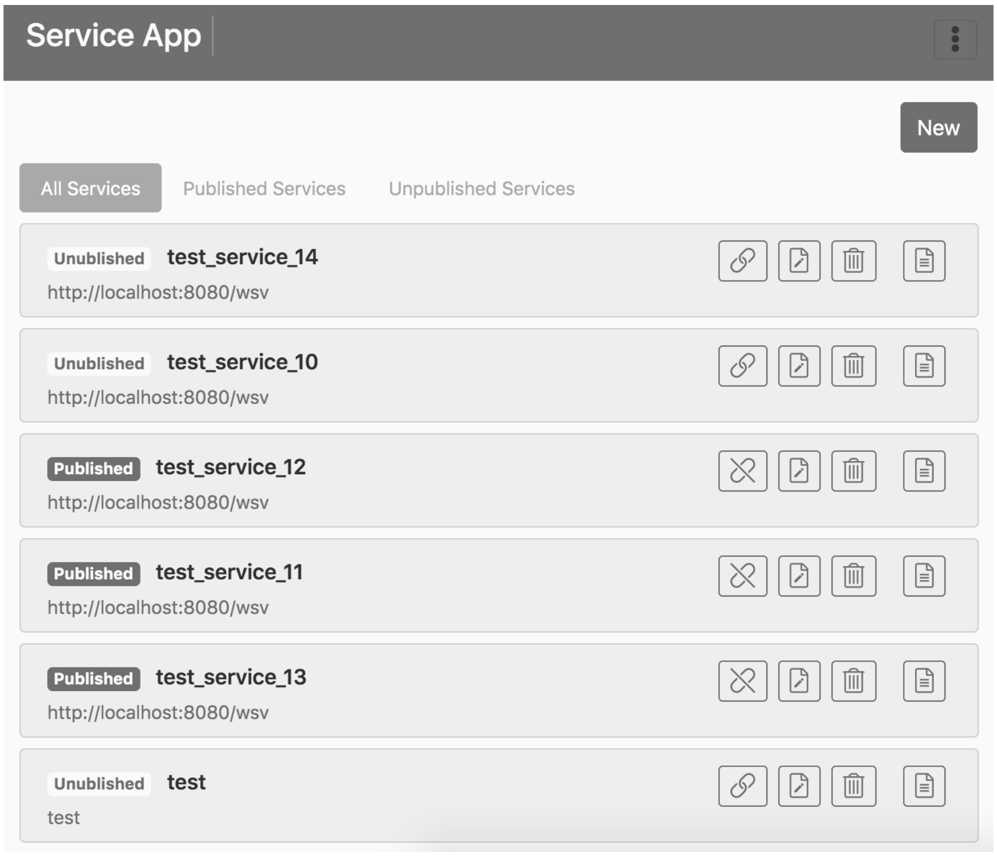
4.3.1 Front-End Layer

Front-end layer is the HTML layer which is used by end users for describing or managing purposes of services. Then it contains CSS3 for styling purposes. Additionally, this layer contains TypeScript for scripting purposes. Main interfaces the solution includes are,

1. Web service management UI
2. User management UI

4.3.1.1 Web Service Management UI

This UI part will provide screen to add, edit, publish, unpublish web services and view logs. Figure 4.1shows the basic layout of the screen.



Unublished test_service_14
http://localhost:8080/wsv

Unublished test_service_10
http://localhost:8080/wsv

Published test_service_12
http://localhost:8080/wsv

Published test_service_11
http://localhost:8080/wsv

Published test_service_13
http://localhost:8080/wsv

Unublished test
test

Figure 4-1 service declaring page

The New button on top of the screen is to add a new service. When user clicks on that button it will forwarded to a new page. Then, there are two main sections in the screen, one it to show published services and other one is to show unpublished services. Users can view logs and unpublish a published service but cannot edit anything of a published service until it is unpublished. URL below each service shows the access point path. Logs button allows user to see the history of the service modification. So it can be used to track changes over time and identify any defects of modifying it.

4.3.1.2 User Management UI

User management screen is a very simple screen. This allows administrator users to add new users to the system, assign to a project or dismissing from a project or group.

4.3.2 JSON Processor Layer

The primary responsibility of this layer is reading, creating and modifying JSON data. Since the service definitions are stored as JSON data, this layer can create initial JSON data string. Then, this layer can modify existing JSON string when users want to update the service definition. Finally, reading the part of the JSON data can be done using this layer. This pars can be general information or service related data such as input parameters, output format, data model format and item count information, etc.

4.3.3 Persistence Layer

The persistency layer will include a relational database which is implemented using MySQL. Figure 4.2 shows the schema diagram for the solution. Since the solution is mainly designed for data processing requirements rather than data savings, the schema diagram is very simple. There are three tables to store service information, user information and group information.

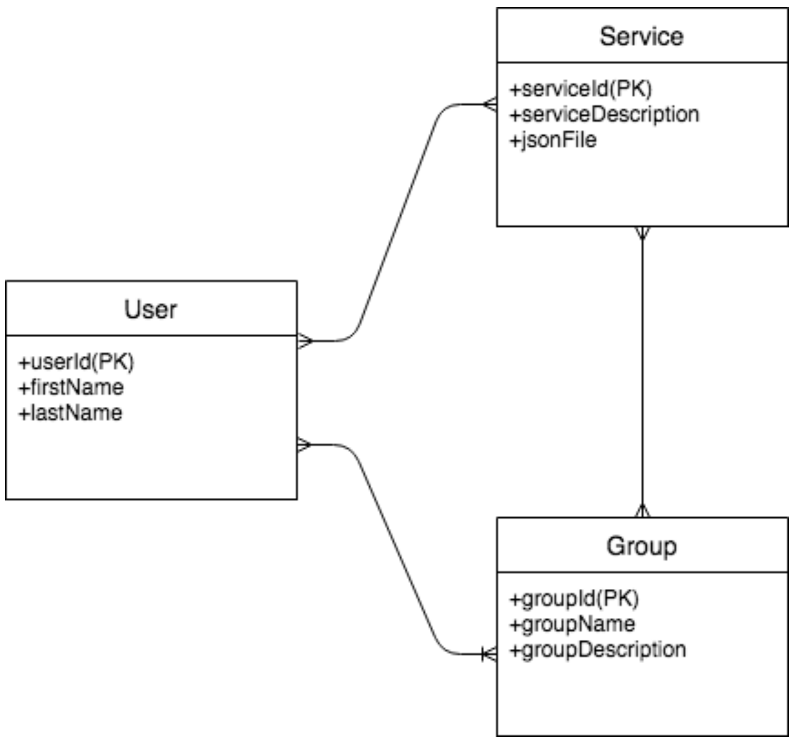


Figure 4-2 schema diagram

User Schema – Contains user information such as profile and login information.

Service Schema – Service related information such as service name description and JSON file or file path.

Group Schema – this table includes user groups information. User groups can be performed project wise or company wise.

One user can have multiple service as well as one service can have multiple users so the relation between them is many-to-many.

Also, one service can have many groups as well as one group can have multiple services. Therefore, the relation between service table and group tables is many-to-many.

User and group table relation table is also many-to-many since one user can have multiple groups and a one group can have multiple users.

4.4 Web service for UI/Core Interaction

The entire solution is designed using client-server architecture. Client part represents user interfaces and client-side logic which is used by end users to define their web services. Then, server-side is the core functionality of the solution which will be described later. Communication between the client and the server is accomplished through a web service. This web service layer allows maintaining a loosely coupled connection between the client and the server.

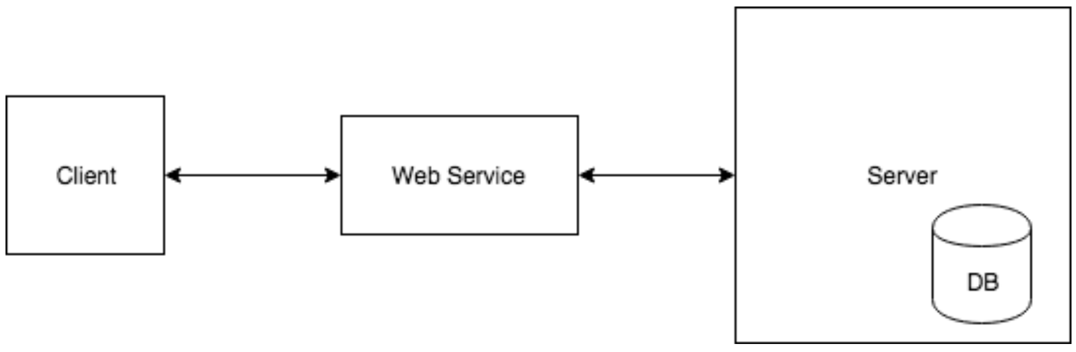


Figure 4-3 user modules and backend

4.5 Data Generation Module Implementation

Main responsibility of this part is generating mock data for requests. There are number of fake data type available but for this product it contains only few of them. Currently supporting data types are Addresses, Book names, Company names, Date and times, Email addresses, Phone numbers, Color names and ID numbers. All those numbers will be generated from a dictionary. There are three major part of this module.

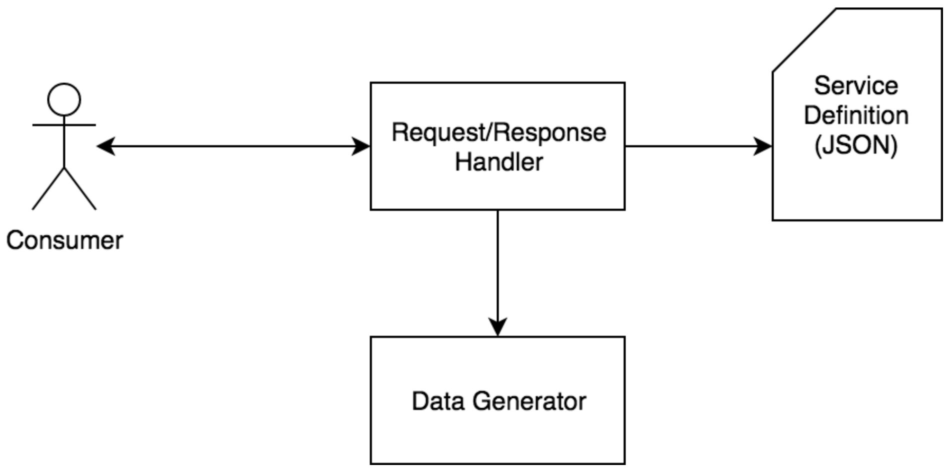


Figure 4-4 Architecture of Data Generation

Request/Response Handler - Requests come from the consumer. Consumer sends those requests when they want to access some data. Request methods can be GET or POST. Request may have parameter in it's body or URL. POST methods will send parameters through the body of the request message as a JSON string. GET method will send parameters as parts of the URL. However, when it comes to this module, everything will be processed in the same manner. Request contains significant data to the data generator. It may contain basic building blocks to build the response such as request method, data type and amount of data.

Service Definition Data – This part includes the data file of the service definition. Format of this data is JSON. When consumer request a data from the system, first it load the correct service definition file from the database. Then it processes it and identify the correct method. Then entire format of the service method is sends to the request/response handle part.

Data Generator – This is where actual fake data is generated. When the request/response handle knows what data is required, it sends request to the data generator with required type.

Then it generates the data and send as a String. If it needs list of data, then multiple request should be sent to the data generation part.

Consumer-Request/Response Handler Communication – Communication between the consumer and the request/response handler will be controlled by servlet requests and responses. Request body will contain JSON data when it is necessary. Response body will also be in JSON format.

4.6 Post Processing of Services

After completing the service definition does not mean that the service is being virtualized. It needs to be published. This publish feature enable the easy management of the services. It can activate and deactivate services. Only published services will be able to access publically.

When a service is being published, simple URL will be generated for each services to identify the services uniquely in the web. This URL is the access point URL of a service. Developers used this URL to access the service. Service access can be restricted by unpublishing.

4.7 Additional Features

There are few extra features even it is not directly connected with service virtualization part. Logs of service modifying history is the most important one. This allows team to see how the service was modified over the time. So tracking any issue is easy. Additionally, it can be used to see how request was come to it and how did it respond with data. Testes can use this information to debug the applications.

4.8 Chapter Summary

This chapter has discussed about the actual implementation strategies of the proposed solution and technology selection for each module. Since the development process is still in progress, most of the high tech items are not discussed. In the final thesis, this chapter will contain comprehensive details of the technical approaches for the solution.

Chapter 5

Evaluation and Testing

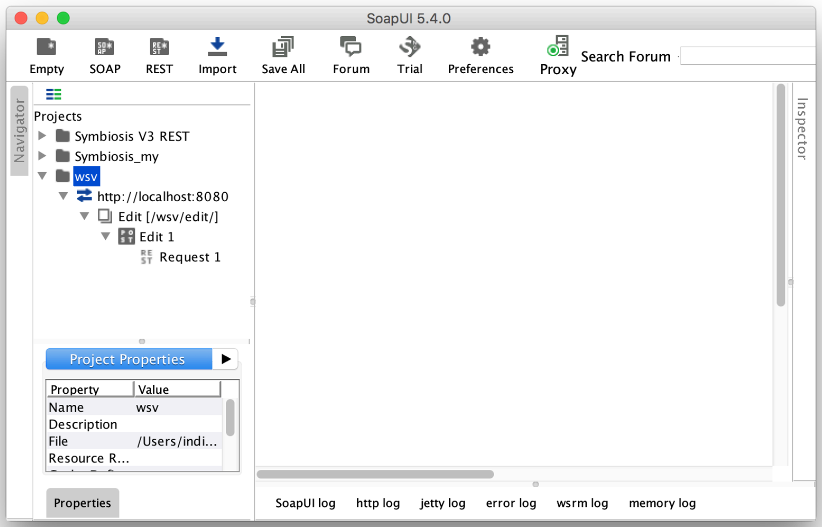
5.1 Introduction

The proposed solution can be evaluated using any industry standard tools which can generate HTTP requests such as SoapUI [21], Postman [22] and Apache JMeter [23]. These tools allow building custom HTTP requests with different user parameters. User parameters can mainly be HTTP method, payload, and the content type. Content type should support JSON since the solution can process only JSON.

SoapUI and Postman are rest and SOAP testing tools. But Apache JMeter is a performance measuring tool. So this can be used to measure time-based and load based testing. For this project, SoapUI will be adequate since its ability to send fully customized HTTP requests.

5.2 Setting up the evaluation environment

SoapUI can be downloaded from its source [1] directly and install it. Basic UI can be seen below after opening the tool.



Property	Value
Name	wsv
Description	
File	/Users/indi...
Resource R...	

Figure 5-1 Basic window of the SoapUI

5.3 Preparing sample services

Sample service can be created by making an empty SOAP project. Then can add requests to that SOAP project.

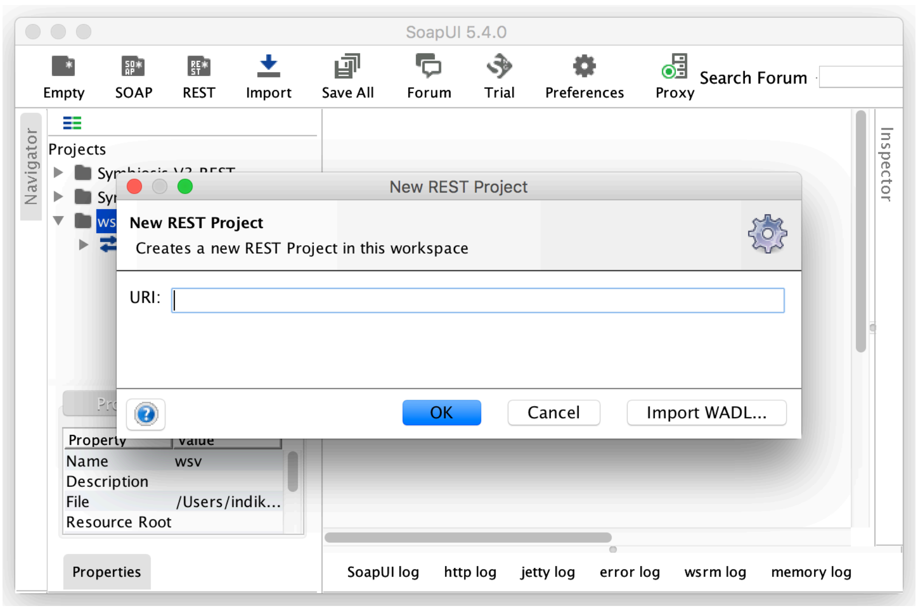


Figure 5-2 Creating a new REST project

Then, add multiple request with different parameter as described in the topic 5.4

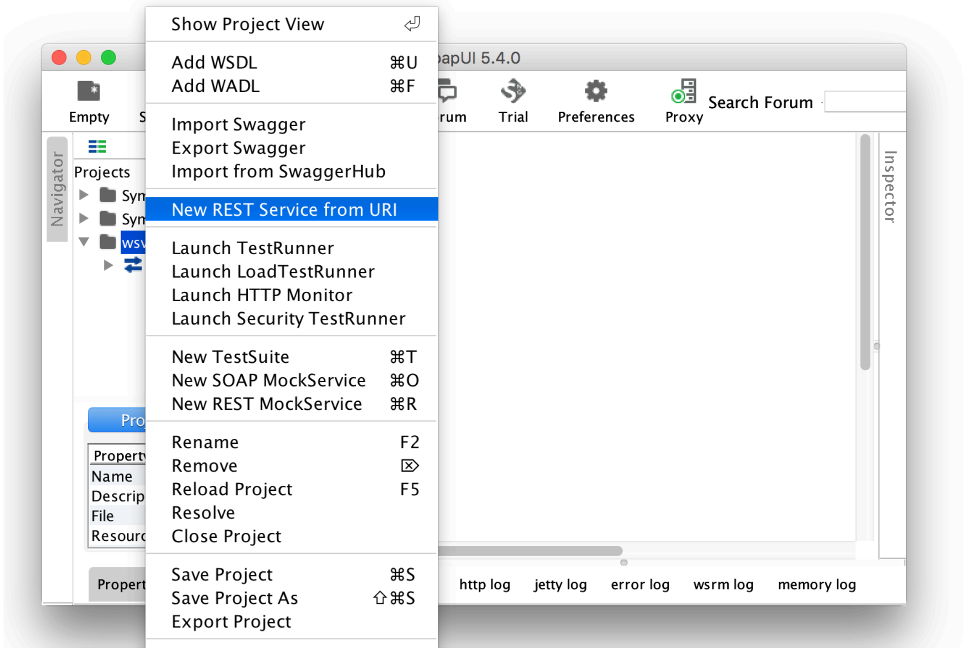
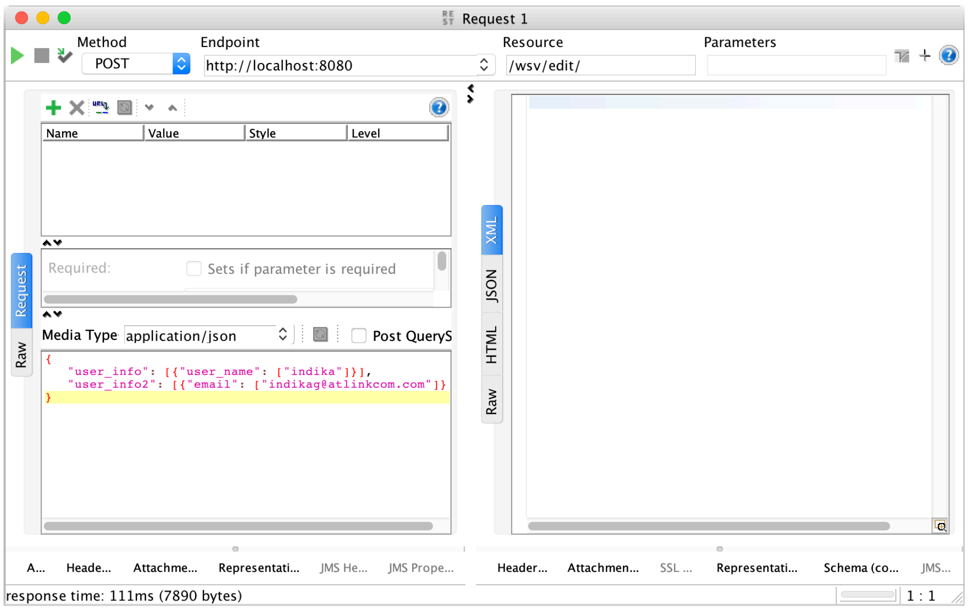


Figure 5-3 Adding REST requests

The request editor window will provide an option to change method, add payload, add header fields and add parameters. Also, it provides a formatted view along with the raw view of both request and response.



Endpoint: http://localhost:8080
Resource: /wsv/edit/
Required: Sets if parameter is required
Media Type application/json
Post QueryS

```
{
   "user_info": [{"user_name": ["indika"]}],
   "user_info2": [{"email": ["indikag@atlinkcom.com"]}
}
```

response time: 111ms (7890 bytes)

Figure 5-4 request editor

5.4 Request generation

Formation of components of main HTTP methods that the system is going to support can be shown as below with mockup data. Request URL represents the endpoint of the service call, Parameters represents the URL parameters and Payload represents the message body. Message body's type should only be "application/json".

POST

Request URL	http://localhost:8080/wsv/serv1/getUserInfo
Parameters	none
Payload	{"user_id":"As3345d"}

Table 5-1 format of the POST request

GET

Request URL	http://localhost:8080/wsv/serv1/getUserInfo
Parameters	?id=23445&ssid=33ee4

Payload	none

Table 5-2 format of the GET request

PUT

Request URL	http://localhost:8080/wsv/serv1/getUserInfo
Parameters	?id=23445&ssid=33ee4
Payload	none

Table 5-3 format of the PUT request

DELETE

Request URL	http://localhost:8080/wsv/serv1/getUserInfo
Parameters	?id=23445&ssid=33ee4
Payload	none

Table 5-4 format of the DELETE request

5.5 Evaluation of results through parameter changes

Evaluation of the solution can be done by creating multiple services and inspecting the possibility of accessing them (virtualization of services) and generating data for each service with different parameters.

5.5.1 Evaluation through service virtualization

Create multiple services through the web interface and add hello world method. Then see whether those are accessible through the SoapUI tool.

Formation of the URL of services can be illustrated as below. Serv1, serv2 etc. are the name of the service which is initially given.

1. http://localhost:8080/wsv/serv1/helloworld
2. http://localhost:8080/wsv/serv2/helloworld
3. http://localhost:8080/wsv/serv3/helloworld
4. http://localhost:8080/wsv/serv4/helloworld

5.5.2 Evaluation through data generation

HTTP body, HTTP parameter and Response types will be modified to evaluate the data generation part.

5.5.2.1 Evaluation of a simple request

A simple query will contain a simple GET request. Initial parameters will be used to generate random data. Below service will get useful information with name, address, and the company name. Different data will be generated for each request.

Service URL – http://localhost:8080/wsv/serv1/getUserInfo

Request Body – empty

Request Parameters – empty

Responses –

<u>Test Case 01</u>

{"users": [{"name": "Caleb Robel IV","company": "Hahn, Hahn and Hahn","address": "Suite 940 215 Carter Isle, Elfriedaberg, CO 40371"}]}

<u>Test Case 02</u>

{"users": [{"name": "Isaiah Hills","company": "Koepp, Koepp and Koepp","address": "368 Pearl Key, Krischester, NE 53945-8525"}]}

<u>Test Case 03</u>

{"users": [{"name": "Lamont Langworth","company": "White-White","address": "168 Mueller Falls, South Winfieldmouth, MI 95595"}]}

5.5.2.2 Changing HTTP method

Defined methods are intended only to generate data for the defined HTTP method.

5.5.2.3 Changing response type

Response type means attribute type such as first name, last name, age etc.

<u>Test Case 01-</u>

Before change the response type

{"users":[{"name":"Kira O'Keefe","company":"Christiansen, Christiansen and Christiansen","address":"Apt. 654 15951 Brendan Crescent, Hectorshire, NE 35309"}]}

After changing company type to date type

{"users":[{"name":"Devonte Block","date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt. 076 03469 Marquardt Common, Lake Reymundofurt, HI 81326-7964"}]}

5.5.2.4 Changing response count

Number of items returns in the response can be change upto any amount. Four test cased are demonstrated here.

Test Case 01 – item count 1

{"users":[{"name":"Devonte Block","date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt. 076 03469 Marquardt Common, Lake Reymundofurt, HI 81326-7964"}]}

Test Case 02 – item count 2

{"users":[{"name":"Raymundo Schimmel","date":"Sat Dec 13 01:26:40 IST 1980","address":"Suite 351 072 Romaguera Isle, West Jaleelborough, ME 53230-6979"},{"name":"Dr. Casper Weimann","date":"Mon May 25 19:52:01 IST 1964","address":"2594 Hagenes Camp, West Florencio, ND 01149"}]}

Test Case 03 – item count 3

{"users":[{"name":"Kenny Hudson","date":"Thu May 27 16:49:05 IST 1993","address":"Apt. 074 980 Lenora Square, Harveyland, NE 18674"},{"name":"Dominique Emmerich","date":"Wed Aug 27 14:24:12 IST 1975","address":"Suite 645 880 Ladarius Hills, Gretchenport, CA 44362-2578"},{"name":"Milan Ledner","date":"Fri Sep 21 15:48:38 IST 1979","address":"Apt. 398 625 Linda Forge, Smithview, ID 62352-3975"}]}

Test Case 04 – item count 4

{"users":[{"name":"Kenny Hudson","date":"Thu May 27 16:49:05 IST 1993","address":"Apt. 074 980 Lenora Square, Harveyland, NE 18674"},{"name":"Dominique Emmerich","date":"Wed Aug 27 14:24:12 IST 1975","address":"Suite 645 880 Ladarius Hills, Gretchenport, CA 44362-2578"},{"name":"Milan Ledner","date":"Fri Sep 21 15:48:38 IST 1979","address":"Apt. 398 625 Linda Forge, Smithview, ID 62352-3975"}, "},{"name":"Justin

Timber","date":"Fri Sep 20 05:48:38 IST 1989","address":"Apt. 398 625 Main Street, George Town, ID 62352-3975"}]}

5.5.2.5 Changing the length of attributes

Attributes are attributes of a response model. In the below evaluation, it concerns variable name length generations. When strings are split into substring, spaces are also considered as a part of the generated name. Therefore, at the end of the name will contain space character.

Test Case 01 – name length is 10 and full length of other fields

```
{"users":[{"name":"Devonte Bl","date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt. 076 03469 Marquardt Common, Lake Reymundofurt, HI 81326-7964"}]}
```

Test Case 02 – name length is 8 and full length of other fields

```
{"users":[{"name":"Devonte ","date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt. 076 03469 Marquardt Common, Lake Reymundofurt, HI 81326-7964"}]}
```

Test Case 03 – name length is 6 and Address length is 10

```
{"users":[{"name":"Devont","date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt.076 03"}]}
```

Test Case 04 – name length is 3 and Address length is 3

```
{"users":[{"name":"Dev","date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt "}]}
```

5.5.2.6 Requesting null value for a specific attribute

Under this part, null values have been requested for a requested attribute

Test Case 01 – Requesting null for name attribute

```
{"users":[{"name":"", "date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt. 076 03469 Marquardt Common, Lake Reymundofurt, HI 81326-7964"}]}
```

Test Case 02 – Requesting null for the address attribute

```
{"users":[{"name":"Devonte Black","date":"Sun Jul 21 16:47:08 IST 1957","address":""}]}
```

5.5.2.7 Requesting random null value for a non-specific attribute

Which attribute should be null will be specified under this test case. Any attribute value can contain null values.

Test Case 01 – Requesting null for any attribute

{"users":[{"name":"", "date":"Sun Jul 21 16:47:08 IST 1957","address":"Apt. 076 03469 Marquardt Common, Lake Reymundofurt, HI 81326-7964"}]}

Test Case 02 – Requesting null for any attribute

{"users":[{"name":"Devonte Black","date":"","address":""}]}

Test Case 02 – Requesting null for any attribute

{"users":[{"name":"","date":"","address":""}]}

5.5.2.8 Requesting hard coded value for a specific attribute

Hard coded value will be thrown always. In this case, attribute value is already known.

Test Case 01 Hard coded user name

{"users":[{"name":"Michel George", "date":"Sun Jul 21 16:47:08 IST 1957","address":"Park Avenue, Houston"}]}

{"users":[{"name":"Michel George", "date":"Sun Jul 03 6:40:23 IST 2007","address":"Marine Drive, California"}]}

5.5.2.9 Requesting hard coded value for a non-specific attribute

This test case is not testable since it cannot be implemented. Hard coded values and attributes are always known.

5.6 Chapter Summary

This chapter has discussed the evaluation and testing strategies for the proposed solution. SoapUI will be used to evaluate the system results since it facilitates to send custom HTTP requests.

Services can be tested via creating multiple services as well as changing parameters of a service. HTTP method, response size and parameters are the main variables considered to evaluate the solution.

Chapter 6

Conclusions and Future Work

6.1 Introduction

In this dissertation, a new way of virtualizing web services to improve the development and testing process has been presented. In this chapter, it will discuss the final conclusions of the presented approach and future direction that some interested party can take to continue to improve web service virtualizing approaches.

6.2 Conclusions

There are many virtualization tools with many features but most of them have less simplicity and hard-coded data returning mechanisms. This type of tools can provide some level of virtualization facility for developers to continue their works and but for testers, this is not advantageous anymore. The proposed solution cares about both parties. It provides a facility to define the output. Defining services cover a large part of the virtualization requirements since it includes how to format the data and how to generate responses as per user requests. So testers also can generate data for testing purposes as well as developers. Predefined data types such as user names, email addresses etc. will provide additional flexibility.

All tools that were studied in the literature review was designed to use by technical persons in the very low level of software development processes. Most of the time, this is going with development or prototype phase. But the given solution is not restricted for the developers or testers even the majority of users can come from these layers. Top level peoples like managers and business analysts can use this during the requirement gathering phase without having deeper knowledge about web service technologies.

6.3 Future work

Importance and usage of web service virtualizing techniques will be increased in the future. Different conditions, requirements, and changes may need to take place in the future to improve the process of service virtualization.

Standalone tools are not more sufficient in the software development context since almost all processes consist of portable and sharable documents, diagrams and sketches which follow industry standards. Hence they are compatible with work with multiple tools. So a new tool should also contain the same capacity while working with other tools. If the proposed solution can provide some way of sharing data in the system with other tools such as project management tools, software architecting tools etc., then the usage of this solution becomes vital to the industry.

Showing a preview while working with some tool always lead to reduce the number of errors that can happen even from experienced users. Photo editing tools and HTML editing tools are some of them. So the users can continue works and look into the preview when they face some confusions. This technique can be used in this tool also to reduce errors and make the users' duty easy. So it should show the final JSON response for a service method with some arbitrary data.

Once a service is defined, it can be modified multiple times from different users throughout the development life cycle. Finally, this can cause a mess since it is hard to identify who did what and when. So keep a track of changes with who did what information can reduce the mess. Additionally, this can be used to maintain a version code for each modification of the service. This version number is important to rollback functionalities.

Most of the time, the client-side script of web services are boilerplates in any programming language except a few lines of codes such as service endpoint etc. Since the solution contains a service definition, it can be used to generate client-side codes up to some level. Or this can be integrated with another service to perform this task by making another intermediate data level between services. So users can download client sides codes in required language and do only necessary changes before merging into their original projects.

6.4 Chapter Summary

This chapter discussed the importance of the web service virtualization approach under the conclusion part and some future works which is not intended to do under this approach.

References

[1] Clancy, T. The Standish Group Report, Retrieved Feb 20, 2019 from https://www.projectsmart.co.uk/white-papers/chaos-report.pdf, Chaos report, 1995.

[2] I. Attarzadeh and S. Hock Ow, "Project Management Practices: The Criteria for Success or Failure", Department of Software Engineering, Faculty of Computer Science & Information Technology, University of Malaya, 2008.

[3] F. Brooks, The Mythical Man-Month: Essays on Software Engineering. Addison-Wesley, 1995, pp. 13-29.

[4] Norshidah mohamed, Batiah Mahadi, Suraya Miskon, Hanif Haghshenas, Hafizuddin Muhd Adnan, "Information System Integration: A Review of Literature and a Case Analysis", 1International Business School, 2Faculty of Computing Universiti Teknologi Malaysia, 2014.

[5] G. Morel, H. Panetto, F. Mayer and J. Auzelle, "System of Enterprise-Systems Integration Issues: an Engineering Perspective", Centre de Recherche en Automatique de Nancy (CRAN - UMR 7039), Nancy-University, CNRS, France, 2009.

[6] T. Bret, Parallel Development Strategies for Software Configuration Management, 12th ed. Rue des Marronniers 25, CH-1800 Vevey, Switzerland: Martinig & Associates, 2004, pp. 2-11.

[7] S. Zunke and V. D'Souza, "JSON vs XML: A Comparative Performance Analysis of Data Exchange Formats", Vishwakarma Institute of Information Technology, University of Pune, India, 2014.

[8] Danut-Octavian SIMION, "Java facilities in processing XML files - JAXB and generating PDF reports", Informatica Economica, Volume 12, Issue 3 2008.

[9] Jackson Documentation, http://wiki.fasterxml.com/JacksonDocumentation

[10] "Traffic Parrot - stubbing, mocking, and service virtualization", Trafficparrot.com, 2019. [Online]. Available: https://trafficparrot.com/. [Accessed: 21- Aug- 2018].

[11 "API Mocking and Service Virtualization Tool | ServiceV Pro", Smartbear.com. [Online]. Available: https://smartbear.com/product/ready-api/servicev/overview. [Accessed: 10- Dec-2018].

[12] "WireMock", WireMock. [Online]. Available: http://wiremock.org/. [Accessed: 22- Dec- 2018].

[13] B. Byars, "mountebank - over the wire test doubles", Mbtest.org. [Online]. Available: http://www.mbtest.org/.

[14] "Hoverfly by SpectoLabs", Hoverfly by SpectoLabs. [Online]. Available: https://hoverfly.io/.

[15] "Service Virtualization: Application & Data Simulation Software | Micro Focus", Micro Focus. [Online]. Available: https://software.microfocus.com/en-us/products/service-virtualization/overview. [Accessed: Dec- 2018].

[16] "Parasof Virtualize - Service Virtualization | Parasoft", Parasoft.com. [Online]. Available: https://www.parasoft.com/products/virtualize. [Accessed: Dec- 2018].

[17] "CA Service Virtualization - CA Technologies", Ca.com. [Online]. Available: https://www.ca.com/us/products/ca-service-virtualization.html. [Accessed: Dec- 2018].

[18] "Fast, easy hosted mock API service | MockLab", Get.mocklab.io. [Online]. Available: http://get.mocklab.io/. [Accessed: Dec- 2018].

[19] "Rational Test Virtualization Server - Overview - India", Ibm.com. [Online]. Available: https://www.ibm.com/in-en/marketplace/rational-test-virtualization-server. [Accessed: Dec- 2018].

[20] "Orchestrated Service Virtualization - Tricentis", Tricentis. [Online]. Available: https://www.tricentis.com/orchestrated-service-virtualization/. [Accessed: Dec- 2018].

[21] "The World's Most Popular API Testing Tool | SoapUI", Soapui.org. [Online]. Available: https://www.soapui.org/. [Accessed: Dec- 2018].

[22] "Postman | API Development Environment", Postman. [Online]. Available: https://www.getpostman.com/. [Accessed: Dec- 2018].

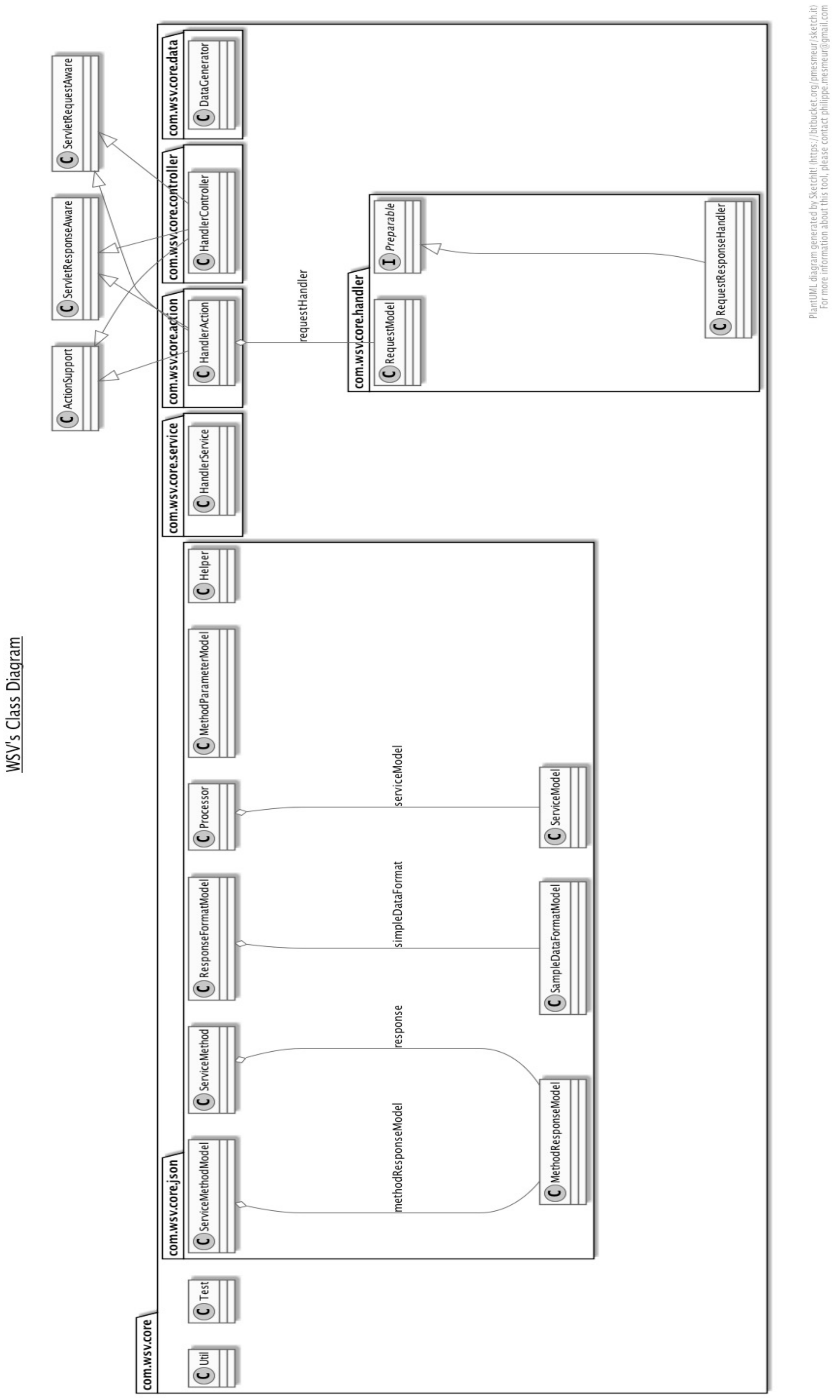
[23] "Apache JMeter - Apache JMeter™", Jmeter.apache.org. [Online]. Available: https://jmeter.apache.org/. [Accessed: 2018].

Appendices

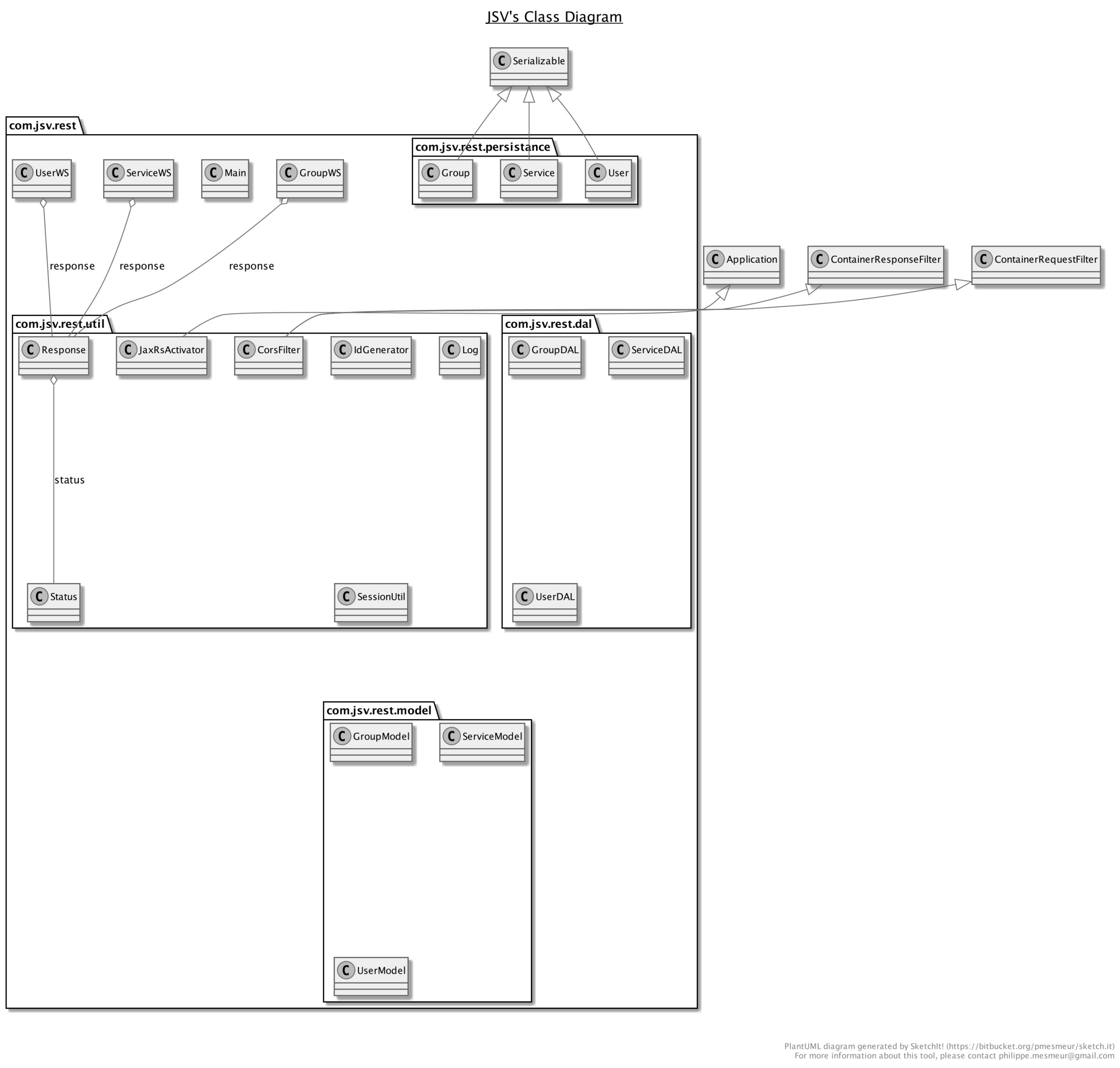
Appendix A – JSON Format

```
{
 "serviceName": "sample_service_users",
 "serviceDescription": "sample service for user management module",
 "serviceMethods": [
  {
   "methodName": "getUserInfo",
   "type": "post",
   "parameters": [
    {
     "parameterName": "userId",
     "parameterValue": "123",
     "parameterType": "String"
    }
   ],
   "response": {
    "minSize": 1,
    "maxSize": 3,
    "name": "users",
    "format": [
     {
      "name": "userName",
      "type": "string",
      "complexDataFormat": null,
      "simpleDataFormat": {
       "type": "name",
       "specific": "no",
       "defaultValue": "Indika Gunawardana"
      }
     },
     {
      "name": "country",
      "type": "string",
      "complexDataFormat": null,
      "simpleDataFormat": {
       "type": "country",
       "specific": "no",
       "defaultValue": ""
      }
     },
     {
      "name": "school",
      "type": "string",
      "complexDataFormat": {
       "minSize": 1,
       "maxSize": 3,
       "name": "school",
       "format": [
            {
              "name": "schoolName",
              "type": "string",
              "complexDataFormat": null,
              "simpleDataFormat": {
                "type": "name",
                "specific": "yes",
                "defaultValue": "Pattalagedara Vidyalaya"
              }
            },
            {
              "name": "schoolLocation",
              "type": "string",
              "complexDataFormat": {
                "minSize": 1,
                "maxSize": 1,
                "name": "cityName",
                "format": [
                  {
                    "name": "province",
                    "type": "string",
                    "complexDataFormat": null,
                    "simpleDataFormat": {
                      "type": "name",
                      "specific": "yes",
                      "defaultValue": "Western Provice"
                    }
                  },
                  {
                    "name": "city",
                    "type": "string",
                    "complexDataFormat": null,
                    "simpleDataFormat": {
                      "type": "name",
                      "specific": "yes",
                      "defaultValue": "Veyangoda_000001"
                    }
                  }
                ]
              },
              "simpleDataFormat": null
            }
          ]
        },
        "simpleDataFormat": null
      }
    ]
  }
 }
]
}
```

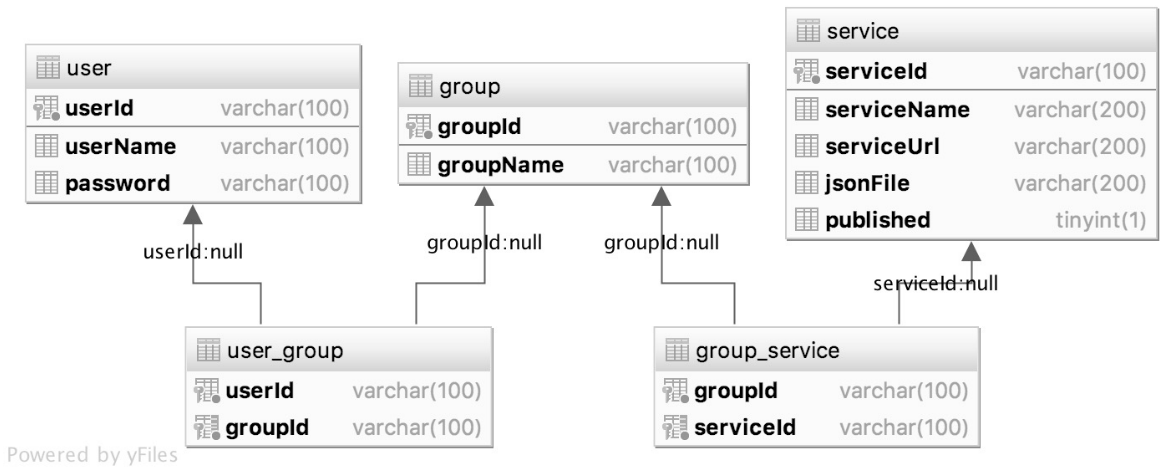
Appendix B – Class Diagrams (Virtualization Core System)



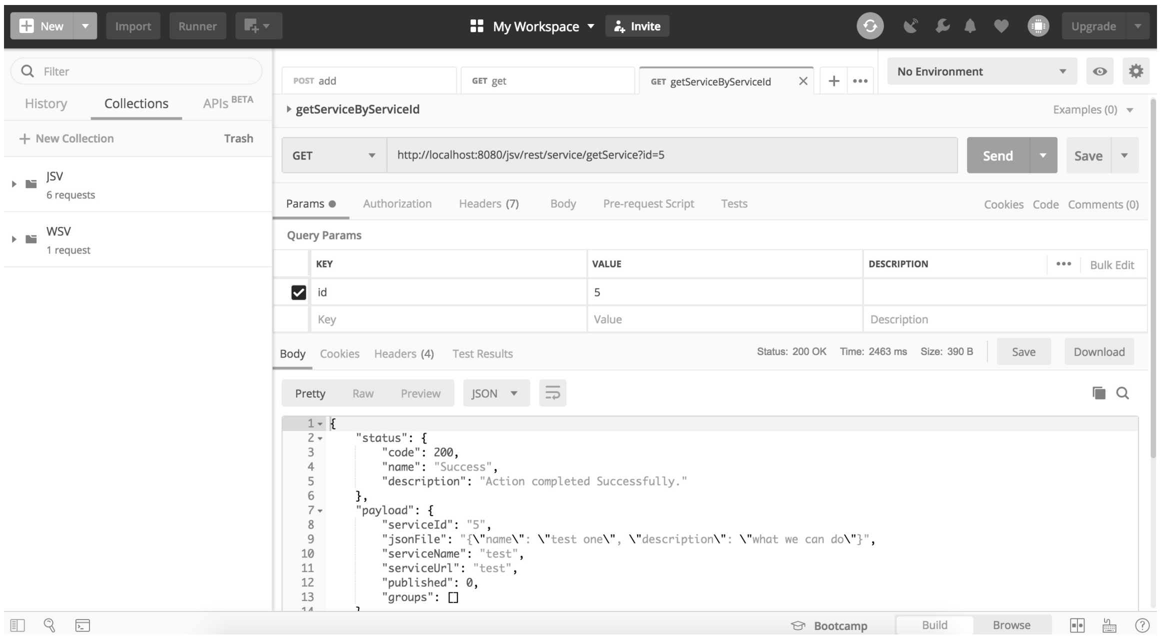
Appendix C – Web Service Class Diagram



Appendix D – ER Diagram



Appendix E – Postman Test Tool



getServiceByServiceId

GET http://localhost:8080/jsv/rest/service/getService?id=5

Query Params

	KEY	VALUE	DESCRIPTION
✓	id	5	
	Key	Value	Description

Status: 200 OK Time: 2463 ms Size: 390 B

```
{
    "status": {
        "code": 200,
        "name": "Success",
        "description": "Action completed Successfully."
    },
    "payload": {
        "serviceId": "5",
        "jsonFile": "{\"name\": \"test one\", \"description\": \"what we can do\"}",
        "serviceName": "test",
        "serviceUrl": "test",
        "published": 0,
        "groups": []
```