International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970)
Volume-3 Number-3 Issue-11 September-2013

Multitenant Software as a Service: Application Development Approach

Suhas Gajakosh[1], Mukta Takalikar[2]

Abstract

Software as a Service (SaaS) is bringing new revolution to IT industry. SaaS has changed the way Software developed, deployed and maintained. SaaS eliminates the requirement of customers (tenants) to purchase, install and maintenance of infrastructure and software. Customers only have to pay for services provided by SaaS vendors. Multitenancy in SaaS application is most important feature for the success of SaaS application. However there are many challenges in the development, deployment, and security of such application. This paper addresses the issue of how to effectively support multitenancy in SaaS application and proposes SaaS architecture to support multitenancy in e-commerce application.

Keywords

Cloud Computing, Software-as-a-Service, Multitenancy, SaaS Architecture, and e-commerce.

1. Introduction

Cloud computing is defined by many researchers, institutes and industries, but the most accepted definition of cloud computing is given by National Institute of Standards and Technology, which is as follows:

NIST Definition of Cloud Computing:
Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models [1]. The basic terminology and methodology to compare the cloud services that best fit for organizational needs are given by Miguel Reixa and et al. in their paper [2]. The authors give basic definitions and comparisons of various cloud computing services and models. The customers of cloud computing subscribes to the service from cloud providers and generally pay-per-use method is used for billing of the subscribed services. This relieves the customers from having on premise hardware or software. The service architecture can be visualized as:

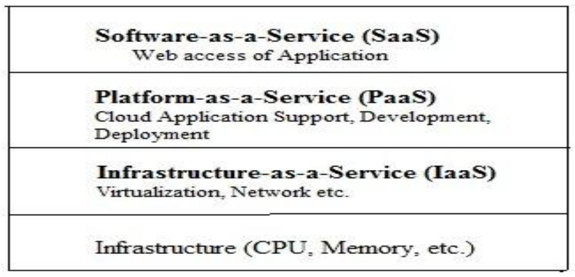
Software-as-a-Service (SaaS)
Web access of Application
Platform-as-a-Service (PaaS)
Cloud Application Support, Development, Deployment
Infrastructure-as-a-Service (IaaS)
Virtualization, Network etc.
Infrastructure (CPU, Memory, etc.)

Figure 1: Cloud Computing Service Architecture

First layer Infrastructure as a service (IaaS) is above the real infrastructure includes various services like virtualization of resources, network infrastructure, etc. The second layer Platform as a service (PaaS)uses the services from IaaS and provides services like platform for cloud application development, VM allocation, etc., and the uppermost Software as a Service (SaaS) layer is used by most of the cloud users, this layer provides the web based applications, business process software (e.g. CRM), etc.

In last decade web has become a platform for the delivery of many software products. This led the pavement for innovation in web based application development of software. Software as a Service is fit to this basic need of new software industry requirement. SaaS is defined by many cloud providers in different words. Here we can define software as a service, a software delivery model which is accessed by its customers through a thin client or web browser and deployed on cloud platform. A customer doesn't need to buy hardware or licensed software to access SaaS application. He only needs to "rent" or "subscribe to" the service and pay according to the agreement between provider and customer, generally they pay-per-use basis for billing the customers.

Suhas Gajakosh, Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India.
Mukta Takalikar, Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India.

Many companies are wishing to convert their web based application to a SaaS solution [3]. Multitenancy is one of the most important characteristic of software as a service. The true multitenant SaaS application is a single instance multiple tenants (customers) model of application delivery i.e. the SaaS application – tenants' shows one-to-many relationship, where a single application (code and database) is accessed by multiple tenants' at a time.

The paper is divided in following manner: 2. Previous Work, 3. Multitenant SaaS Application Model, 4. Conclusion and Future Work, and References.

2. Literature Review

A lot research has been done in how multitenancy can be achieved in SaaS application. A business providing SaaS application cannot be successful without multitenancy. Multitenancy is an organization approach for SaaS application. Key characteristics of multitenancy can be given as 1. Hardware resource sharing, 2. High degree of configurability, 3. Shared application and database instance [4].

Jinan Fiaidhi et al. [5] proposed general multitenancy cloud architecture. The paper talks about managing multitenant data. The approaches that they have used are 1. Storing tenant data in separate databases, 2. Housing multiple tenants in the same database, with tenant specific schema, and 3. Using the same database and same set of tables to host multiple tenants' data.

A document of MSDN Microsoft by Frederick Chong and et al. [6] states architecture for multitenant data.

The multitenancy can be classified according to its tenants' data separation in application database. This classification given in this document can be shown as,

- Separate Databases, This approach uses a separate database for each tenant, a simple but not a true multitenant approach.
- Shared Database, Separate Schemas, This one is applicable approach for data separation and sharing database, this approach is also known as semi-multitenancy approach.
- Shared Database, Shared Schema, This one is a highly referred approach and called as pure-multitenancy approach, in this approach each tenant data is stored in the same database and same schema, the separation of tenant is done by assigning tenant_id to rows in tables that the tenant owns.

3. Problem Analysis

A multitenant SaaS application has to be designed carefully as it includes the shared application and database. The problem of designing an architecture that implements multitenant application is important. This architecture will include the suggestions to choose the database and schema from [6], choose how the system identifies the tenant and how the logic is handled by the system. The architecture to implement multitenant application is less discussed in literature.

Cor-Paul Bezemer and et al. proposed a multitenant architecture conceptual blueprint [4] to implement multitenant SaaS application. Their architecture includes a tenant specific authentication to authenticate tenants, a configuration component that can be implemented to allow customization of multitenant application for each tenant and a database. The authentication authenticates the tenants and it separates the configuration component for each tenant. The configuration component includes layout style, workflow, general configuration etc. to be used for customization by each tenant and the database includes some layers so that it isolates the data in same environment for each tenant.

However the above architecture has some disadvantages related to authentication and database. Their authentication uses a ticket server to identify the tenant, which is not necessary if we use a proper method to implement tenant system e.g. use of subdomain to identify tenant can be used, which we will see in later section. The database uses various layers to isolate tenant specific data which is not actually needed if we use the any method given in [6] by Frederick Chong and et al.

Rouven Krebs and et al. in [7] have proposed various approaches for resource sharing in cloud environment and also defines multitenancy in cloud environment. The papers differentiate in sharing of resources at code base, sharing at data center, sharing using virtualization at different layers of cloud computing.

The white paper by Jason Meiers and et al. of IBM [8] have explained the requirement of multitenant SaaS applications also it gives a various requirements for building multitenant network topology, multitenant databases etc.

Bikram Sengupta and et al. [9] proposed a multitenancy re-engineering pattern, giving the possible sharing of database, configurability of user interface, workflow etc. We are extending our multitenant SaaS application architecture from [4] and [6].

Craig D Weissman and et al. discussed multitenant internet application development platform in [10]. This paper describes the design of the force.com multitenant application.

This clears our problem of multitenant SaaS application for implementation. The next section proposes a solution to above mentioned problems in implementing multitenant SaaS application.

4. Multitenancy SaaS Application Architecture Approach

Before considering architecture approach we will discuss the application that we are going to implement using this approach. We are interested in implementing an e-commerce application which includes products, carts, users and orders as component and tenant metadata of each tenant. This multitenant application is implemented using ruby on rails framework and postgresql as database. This application will be deployed to Heroku [11] (Cloud Platform for rails) and will be accessed by users of various tenants through internet using browsers.

The above diagram shows application architecture for the implementation of multitenant SaaS application in cloud environment. The architecture is divided into four main parts. Each of these parts is designed to support multitenant SaaS e-commerce application.

4.1 Subdomain URL

In general case of simple web application URL determines the address of the application stored. In such a case subdomains are used to identify the component of a main application. These subdomains are handled by DNS servers to locate the server where actual application is stored.

In our case the subdomain is a parameter of the tenant which is used to locate the store of a particular tenant. This identifies the tenant_id of a tenant and all operations in this store are carried out according to the application logic. This system supports multiple tenants with multiple subdomains for a single domain.

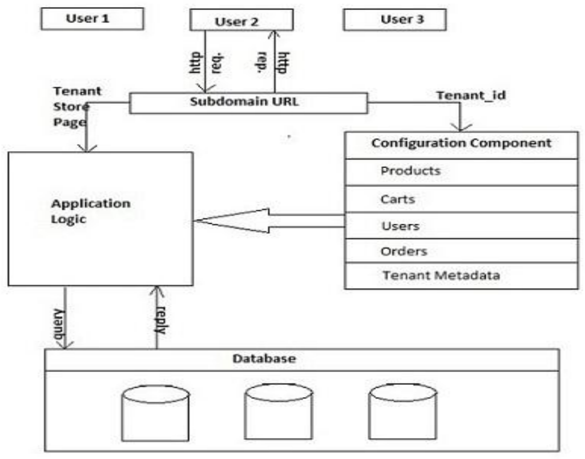

Figure 2: Multitenant SaaS e-commerce Application Architecture

4.2 Application Logic

The application logic is a main module to run the application. This module contains the real code of application, which is being used by each tenant and its users after login to logout. The application logic will include the functionality of e-commerce application. This module is implemented using ruby on rails framework. The application logic is continuously running on the web server in cloud environment. It accepts the http requests from users of the system and replies with html file.

4.3 Configuration Component

In multitenant application configuration per tenant is the most important module of the multitenant application. Our e-commerce application includes various components like products, carts, users, orders etc. These components are scoped according to tenants and identified using tenant_id in components database. In our application products and users are the main configurable components. Each tenant user (seller) is allowed to access configuration components of the system. The user can access the components that belongs to its tenant only and is not able see other tenants components. The tenant metadata includes the tenant_id, name of the tenant and subdomain.

4.4 Database

Database handling of each tenant has some issues related to storage and access to the data. However in particular to the multitenant database Jinan Fiaidhi and et al. in [5] discussed three approaches to implement multitenant database, they are separate databases, shared database - separate schema (semi-multitenancy) and finally shared database - shared schema (pure multitenancy). Also another paper by Stefan Aulbach and et al. discussed multi-tenant SaaS application database and schema mapping techniques in their paper [12]. In our dissertation we have implemented pure multitenancy approach in which tenant_id is shared by all configurable components tables and they are accessed by identifying tenant_id from the subdomain of the tenant. The pure multitenancy scheme isolates the data of each tenant hence providing privacy between each tenant

5. Results

The results of multitenant application can be given in the form of snapshots of store of various tenants and comparing them. We have taken results by running our application on local environment.

In case of single tenant e-commerce application, it can have only one store to purchase or sell products. In our multitenant e-commerce application we have 2 tenants selling their products from their respective store shown using different tenants.

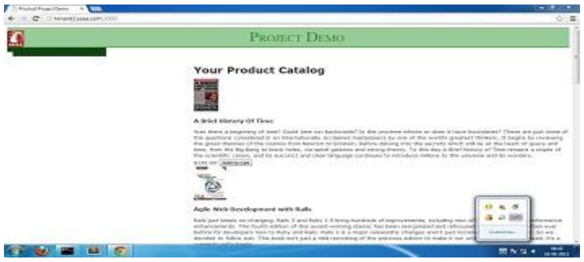

Figure 3: View of Tenant 1 System with its subdomain

The above figure shows a store of tenant1 which has subdomain ‘tenant1’ for domain ‘saas.com’. The above diagram represents the book shop as one of the two tenants.

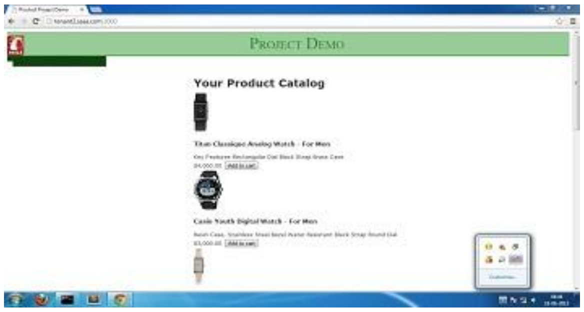

Figure 4: View of Tenant 2 System with its Subdomain

The above figure shows a store of tenant2 which has subdomain ‘tenant2’ for domain ‘saas.com’. The above diagram represents the watch shop as one of the two tenants.

6. Conclusion and Future Work

The paper suggests a new approach for the multitenant SaaS application implementation it also describes the procedure for the implementation of e-commerce multitenant SaaS application. This approach identifies tenant from subdomain URL, providing configurability in multitenancy and most important a data separation for each tenant in cloud environment. This architecture can be used to implement any other multitenant SaaS application.
The architecture can be modified according to the need of application to implemented. A dashboard can be designed to make the tenant able to choose the configurable components for his applications in multitenant application. This will make the multitenant application highly configurable.

References

[1] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, NIST Special publication 800-145, September 2011.
[2] Miguel Reixa, Carlos Costa, Manuela Aparicio, “Cloud Services Evaluation Framework”, OSDOC’12 June 11, 2012.
[3] Scott Chate, “Convert your web application to a multitenant SaaS solution”, White paper, Developer-Works, IBM, 14 Dec 2010.
[4] Cor-Paul Bezemer, Andy Zaidman, “Multitenant SaaS Applications: Maintenance Dream or Nightmare?” Delft University of Technology Software Engineering Research Group Technical Report Series, September 2010.
[5] Jinan Fiaidhi, Irena Bojanova, Jia Zhang and Liang-Jie Zhang, “Enforcing Multitenancy for Cloud Computing Environment”, IT Pro January/February 2012.
[6] Frederick Chong, Gianpaolo Carraro, and Roger Wolter, “Multitenant Data Architecture, Microsoft Corporation”, Available http://msdn.microsoft.com/enus/library/aa479086.aspx, as on April 2013.
[7] Rouven Krebs, Christof Momm and Samuel Kounev, “Architectural Concerns in Multitenant SaaS Applications.”
[8] Jason Meiers, “Best practices for cloud computing multi-tenancy”, White paper, IBM, 06 Jul 2011.
[9] Bikram Sengupta, Abhik Roychoudhury,

"Engineering Multitenant Software-as-a-Service Systems", ICSE'11, May 21–28, 2011.

[10] Weissman, C. D. and Bobrowski, S. (2009), "The design of the force.com multitenant internet application development platform", In Proceedings of the 35th SIGMOD international conference on Management of data, SIGMOD '2009.

[11] Getting Started with Rails 3.x on Heroku, Available https://devcenter.heroku.com/articles/rails3, as on April 2013.

[12] Stefan Aulbach, Torsten Grust, Dean Jacobs, Alfons Kemper, Jan Rittinger, "Multitenant Databases for Software as a Service: Schema-Mapping Techniques", SIGMOD'08, June 9–12, 2008.

Suhas Gajakosh. Born in Pandharpur, Maharashtra, India on 10th oct. 1988. The author has completed BE in Computer Science and Engg. From Solapur University, and currently pursuing his ME from PICT, Pune University, Pune.

Mukta Takalikar. Born in Nagpur, Maharashtra, India on 07th June 1972. The author has completed M.E. Computer Engineering from C.O.E.P. Pune, B.E. from S.G.G.S.C.O.E & T. Nanded P.G.D.B.M. from BHAVAN's, Mumbai and currently pursuing Ph.D.in Computer Engineering. Currently Working as an Assistant Professor in Computer Engineering, PICT, Pune.