

SaaSpia Platform: Integrating and Customizing On-Demand Applications Supporting Multi-tenancy

Kyoungyun PARK, Heesun WON, Sungjin HUR

SW Service Research Team, Electronics and Telecommunications Research Institute, Korea

hareton@etri.re.kr, hswon@etri.re.kr, sjheo@etri.re.kr

Abstract— SaaS applications support multi-tenancy and dedicated service environments by providing multiple tenants with user interface for customizing their own service. On the other hand, existing web-based applications do not support multi-tenancy and configurability. Therefore, transforming existing web applications into SaaS applications and serving them to multiple tenants is a great challenge. In this paper, we introduce SaaSpia, the general purpose SaaS Platform supporting multi-tenancy and configurability. We also introduce SaaSpia Integration Tool and SaaSpia Configuration Tool, the SaaSpia platform tools to integrate and customize applications.

Keywords— Software as a Service(SaaS), Metadata-driven architecture, Multi-tenancy, Configurability

I. INTRODUCTION

Software as a Service(SaaS) is a service delivery model in which service providers provide clients with softwares in the form of services. SaaS applications serve a big amount of clients in a single instance environment(multi-tenancy) and provide a number of clients with dedicated services in which their service requirements are reflected(configurability).

Configurability is an ability to adapt applications without writing new programming code or recompiling applications in order to reflect requirements of tenants. Therefore clients can easily tailor their applications without developer's help. Multi-tenancy is an architecture that a single instance that is running on a server serves multiple clients. Service providers can reduce resource and management cost by choosing multi-tenancy architecture because multi-tenancy share hardware and software resources.

To accomplish multi-tenancy and configurability, SaaS applications should manage configuration information of tenants as metadata and a SaaS platform should be based on metadata-driven architecture to support these functions. In the metadata-driven architecture, a runtime engine loads the tenant-related metadata and dynamically generates the service when tenant users request a service. This architecture makes it possible to serve tenant-specific services to multiple tenants in the single instance environment.

On the other hand, Existing web applications do not support configurability and multi-tenancy. Therefore in order to transforming web applications into SaaS applications, we

should consider many challenges including software architecture.

In this paper, we describe how to integrate web-based applications on the SaaSpia platform, the general purpose SaaS platform. We also present two types of tools, SaaSpia Integration Tool and SaaS Configuration Tool that are tools for integrating and customizing applications in the multi-tenant environment.

SaaSpia Integration Tool is an online application development tool which enables application developers to integrate on-demand applications into SaaS platform without considering configurability and multi-tenancy.

Tenants can also customize applications using SaaS Configuration Tool. In this paper, we also describe the functionality of the configuration tool to provide clients with configurability such as user interface configuration, business logic configuration, and data schema configuration.

This paper is organized as follows. In the next section, we introduce SaaSpia Platform that is the general purpose SaaS platform developed by ETRI. In section 3, we describe SaaSpia Integration Tool and discuss how to integrate web applications on SaaSpia platform. In section 4, we present SaaS Configuration Tool and show how a tenant administrator customizes SaaS applications using SaaS Configuration Tool. Finally we conclude our paper with future work.

II. SAASPIA: THE GENERAL PURPOSE SAAS PLATFORM

SaaSpia is a general purpose SaaS Platform that supports multi-tenancy and configurability. Figure 1 shows a concept of SaaSpia platform that supports multi-tenancy. A SaaSpia-based application provides tenants with the common service at the first time. Tenants have their own service requirements and want to reflect these requirements in their service. SaaSpia allows tenants to reflect their service requirements through configuration user interfaces. Each tenant can customize user interface, business logic, and data schema in the configuration environment. SaaSpia platform transforms configuration data of tenants into metadata. Finally, SaaSpia platform runs an application in a single instance and provides multiple tenants with a dedicated service for each tenant.

SaaSpia consists of Configurator, Dynamic Runtime Engine, Metadata Manager, Security Manager, and Platform

Management Tool. The role of each module is as in the following.

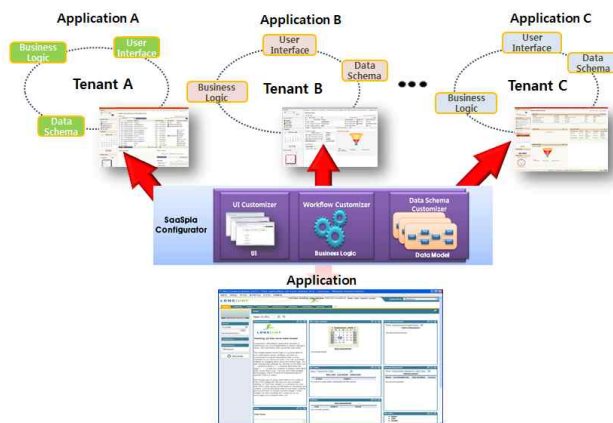


Figure 1. SaaS supporting Multi-tenancy

A. Configurator

Configurator provides tenants with user interface for customizing their service. A tenant administrator can customize user interfaces(web pages), JavaScript-based business logic, and data schema. Web pages are implemented in ExtJS and Configurator supports tree-based user interface model and UI components. When a tenant administrator customizes UI pages and saves them, configurator disassembles UI pages into UI components and stores page information and UI components into databases.

Configurator also provides a tenant administrator with fixed columns and flexible columns. The tenant administrator cannot modify fixed columns because fixed columns are commonly served to all tenants. To customize data schema, the tenant administrator uses flexible columns. Configurator supports user interface to customize flexible columns as a data schema configuration.

When a tenant administrator adds a new edit box or grid column, a new business logic is also required as well as new UI components. Configurator makes it possible to add a simple business logic by providing UI component properties.

B. Dynamic Runtime Engine

Dynamic runtime engine creates a new service dynamically and delivers it to tenant users when tenant users request a service. To compose the service dynamically, runtime engine loads UI components through Metadata Manager and creates UI tree model. The new UI tree model is rendered into a ExtJS based-web page.

After rendering the web page, runtime engine retrieves tenant data. Runtime engine sends tenant SQL statements and receives SQL results from Metadata Manager.

C. Metadata Manager

Metadata Manager supports tenant data isolation and data schema configuration.

Runtime Engine sends a tenant query statement to Metadata Manager. Metadata Manager interprets the tenant query and

translates the tenant query into a target query statement. The target query statement includes tenant identification and flexible columns of the tenant. After retrieving result data, Metadata Manager sends the results to Runtime Engine.

Figure 2 shows the architecture of Metadata Manager. Metadata Manager provides metadata API and tenant data API to support data schema configuration and tenant data isolation.

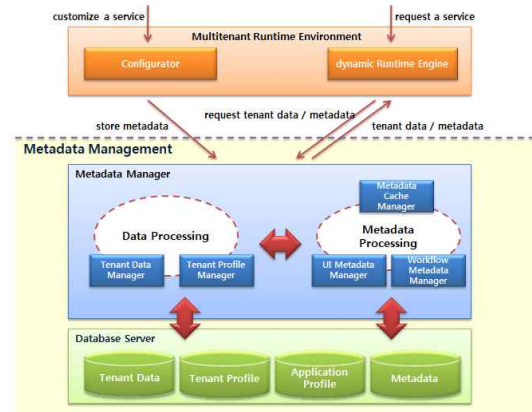


Figure 2. Metadata Manager Architecture

D. Security Manager

SaaS supports role-based access control and component-based access control for security.

Each role has accessible URLs and it is assigned to tenant users. Tenant users can have multiple roles. Each UI component has CRUD privileges. Each role can have CRUD privileges of the UI component. If the tenant user has the role that has Create privilege of a edit box, the editor box is activated when the editor box has Create privilege. If the editor box does not have Create privileges, the editor box is inactivated.

Security Manager checks if the request URL is valid when the tenant user requests a service. If the URL is accessible, Security Manager also decides to activate UI components when Runtime Engine are rendering the web page.

E. Platform Management Tool

Platform Management Tool is monitoring usage of platform resources by tenants and status of services. Platform Management Tool is runned by a platform manager and controls all tenants and services as well as hardware resources.

III. INTEGRATING SAAS APPLICATIONS

Existing web applications are based on single tenant architecture. Therefore they do not support multi-tenancy and configurability. SaaS makes it possible to transform existing web applications into SaaS applications by providing SaaS Integration Tool.

Figure 3 shows the functionalities of SaaS Integration Tool. To be integrated on SaaS, user interface of web applications should be implemented using ExtJS, one of the JavaScript Framework. They should be also based on Spring

framework. ExtJS-based User Interface is transformed into UI metadata and stored into database through SaaSapia Integration Tool.

Integration Tool manages tenant data schema in the form of fixed columns and flexible columns. Fixed columns are data fields that are commonly served to all tenants and flexible columns are data fields that are used to extend tenant data schema for each tenant. Generally application tables do not have flexible columns. Therefore, a service developer should add flexible columns to all application tables. The number of flexible columns is not fixed and it depends on how many flexible columns tenants use.

Business logic is integrated into SaaSapia packages without being transformed if business logic is implemented in Spring framework.

data types by each tenant because flexible column configuration is managed in the form of metadata.

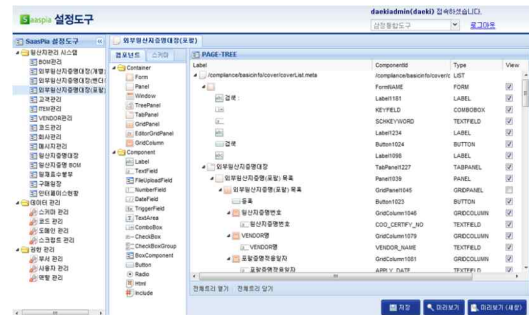


Figure 4. SaaSapia Configuration Tool

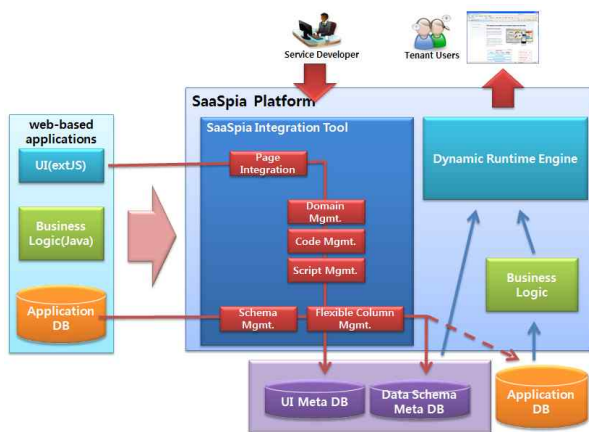


Figure 3. Integrating web applications using SaaSapia Integration Tool

IV. CUSTOMIZING TENANT SERVICE

SaaSapia platform provides tenants with configurability through SaaSapia Configuration Tool.

Figure 4 shows user interface of SaaSapia Configuration Tool. SaaSapia Configuration Tool consists of application page(menu) management, data management, access control management. Application page management has application pages and provides tree-based user interfaces. A tenant administrator can customize user interface by UI component unit. Each UI component provides a property window and an administrator can modify UI component properties through the property window. Customized pages are stored into a database in the form of metadata.

Data management provides mapping information between a user interface and a database. Data management includes schema management, code management, and domain management, and script management.

Schema management provides application schema information including fixed columns and flexible columns. A tenant administrator can customize data schema by choosing a flexible columns and assigning data type to the flexible column. Each flexible column can be assigned to different

Code management deals with master codes and code values that are commonly used in applications. SaaSapia Configuration Tool provides default codes and tenant administrators can add their own master code and code values.

Domain management is used to define data type of UI components, especially grid columns that are mapping into tenant table columns. Domain can be one of basic database data types, codes, and scripts.

Script management is a collection of JavaScript programs that are mainly used to check data validation or execute simple operations. JavaScript is also stored into databases and tenant administrators can add their own JavaScript.

Access Control Management includes department management, user management, and role management. SaaSapia uses role-based access control for security. Therefore tenant users have some roles and can access application URLs that their roles allow.

Configuration by tenants divides into three categories, user interface, business logic, and data schema as follows.

A. User Interface Customization

User Interface is implemented in ExtJS and is based on tree model that its root node is a page URL. A Tenant administrator can customize user interface by modifying a UI tree. Each UI component that consists of pages has properties such as label, layout, size, script, and so on.

B. Data Schema Customization

SaaSapia provides application schema information for schema management. An application schema consists of fixed columns and flexible columns. Fixed columns are table fields that all tenants are commonly using and flexible columns are fields that each tenant is using as dedicated table fields.

To add a table column, tenant administrator chooses a table and adds a flexible field. The flexible field data type is defined using domain. Domain can be basic data type, code, or script. The flexible field chosen by the tenant appears in schema tab in SaaSapia Configuration Tool and the tenant administrator can map the flexible field into a UI component.

C. Business Logic Customization

Business logic is processing values that are retrieved from databases and delivering results. If tenant administrator adds a new UI component such as an editor box or a grid column, the administrator should add new business logic to present a value in the UI component.

Business logic customization can be applied in the JavaScript level and in the java controller level. If business logic that an administrator wishes to add is simple, new business logic can be easily written and added in the JavaScript level by using UI component properties. If more complex business logic is required or a new SQL should be added, an administrator should write a new java controller and store in the databases. The stored business logic is executed by dynamic runtime engine when a tenant requests a service.

V. CONCLUSION AND FUTURE WORK

In this paper, we introduce SaaS pia platform and describe main functions of SaaS pia Integration Tool and Configuration Tool. SaaS pia Integration Tool is used for integrating web applications and SaaS pia Configuration Tool is used for customizing Services. SaaS pia Integration Tool is a tool for service developers and SaaS pia Configuration Tool is a tool for tenant administrators. SaaS pia Integration Tool provides common services to all tenants and SaaS pia Configuration Tool provides the dedicated service to the specific tenant. SaaS pia platform supports SaaS maturity Level 3 and we are developing PaaS (Platform as a Service).

ACKNOWLEDGMENT

This work was supported in part by the IT R&D program of MIS/IITA [2009-S-033-01, Development of SaaS Platform for Software Service of Small and Medium Enterprises, and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology [20100022431].

REFERENCES

- [1] K. Selçuk Candan, Wen-Syan Li, Thomas Phan, Minqi Zhou, "Frontiers in Information and Software as Services," *icde*, pp.1761-1768, 2009 IEEE International Conference on Data Engineering, 2009
- [2] CJ Gui, W Sun, Y Huang, ZH Wang, B Gao, "A Framework for Native Multi-Tenancy Application Development and Management", *E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, 2007. CEC/EEE 2007. The 9th IEEE International Conference on, 23-26 July 2007
- [3] Nitu, "Configurability in SaaS (Software as a Service) Applications", *ISEC*, Pune, India, Feb. 2009
- [4] CP Bezemer, A Zaidman, "Multi-Tenant SaaS Applications: Maintenance Dream or Nightmare?", *IWPSE-EVOL*, Antwerp, Belgium, Sept. 2010
- [5] CD Weissman, S Bobrowski, "The Design of the Force.com Multitenant Internet Application Development Platform", *SGIMOD*, Providence, Rhode Island, July 2009.
- [6] YV Natis, "Reference Architecture for Multitenancy: Enterprise Computing in the Cloud", *Gartner*, Dec. 2008
- [7] Wei Sun, Xin Zhang, Chang Jie Guo, Pei Sun, Hui Su, "Software as a Service: Configuration and Customization Perspectives", In *Proc. of the IEEE Congress on Services Part II*, 2008, pp. 18-24.